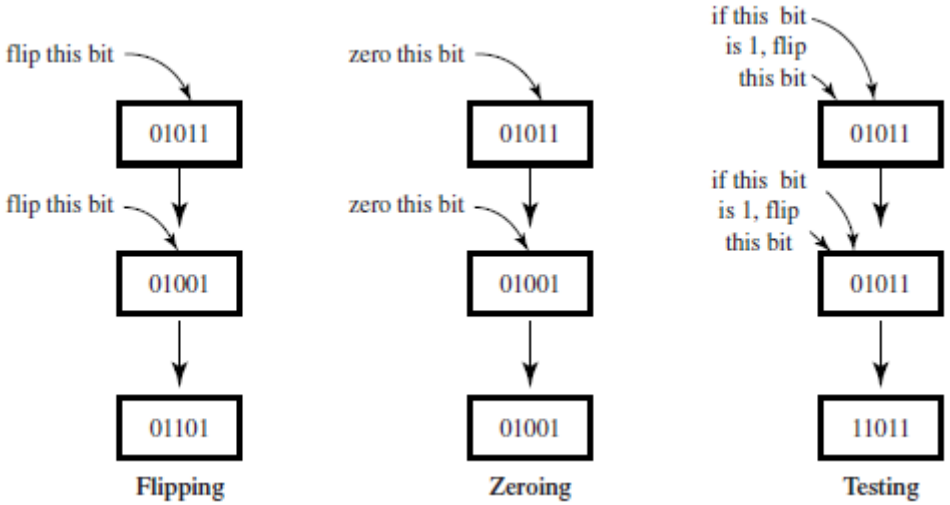
Wstęp i przegląd historyczny, czyli o co w tym wszystkim chodzi?

Komputery to niesamowite maszyny. Wydają się być w stanie zrobić wszystko. Latają samolotami i statkami kosmicznymi, kontrolują elektrownie i niebezpieczne zakłady chemiczne. Firmy nie mogą już działać bez nich, a coraz więcej skomplikowanych procedur medycznych nie może być wykonywanych pod ich nieobecność. Służą prawnikom i sędziom, którzy szukają precedensów sądowych w dziesiątkach udokumentowanych procesów, a także pomagają naukowcom w wykonywaniu niezwykle skomplikowanych i wymagających obliczeń matematycznych. Przekierowują i kontrolują miliony połączeń telefonicznych w sieciach obejmujących kontynenty. Wykonują zadania z niezwykłą precyzją - od czytania map i składu po graficzną obróbkę obrazu i projektowanie układów scalonych. . Mogą odciążyć nas od wielu nudnych obowiązków, takich jak skrupulatne śledzenie wydatków domowych, a jednocześnie zapewnić nam urozmaiconą rozrywkę, np. gry komputerowe lub skomputeryzowana muzyka. Co więcej, dzisiejsze komputery ciężko pracują, pomagając zaprojektować jeszcze potężniejsze komputery jutra. Tym bardziej niezwykłe jest to, że komputer cyfrowy – nawet ten najnowocześniejszy i najbardziej skomplikowany – może być traktowany jako po prostu duża kolekcja przełączników. Te przełączniki lub bity, jak się je nazywa, nie są „odwracane" przez użytkownika, ale są specjalnymi, wewnętrznymi przełącznikami, które są „odwracane" przez sam komputer. Każdy bit może znajdować się w jednej z dwóch pozycji lub, inaczej mówiąc, może przyjmować jedną z dwóch wartości, 0 lub 1. Zazwyczaj wartość bitu jest określona przez pewną charakterystykę elektroniczną, na przykład, czy dany punkt ma ładunek dodatni lub ujemny. Komputer może bezpośrednio wykonać tylko niewielką liczbę niezwykle trywialnych operacji, takich jak odwracanie, zerowanie lub testowanie. Odwracanie zmienia wartość bitu, zerowanie zapewnia, że bit kończy się na pozycji 0, a testowanie robi jedną rzecz, jeśli bit jest już na pozycji 0, a drugą, jeśli tak nie jest.

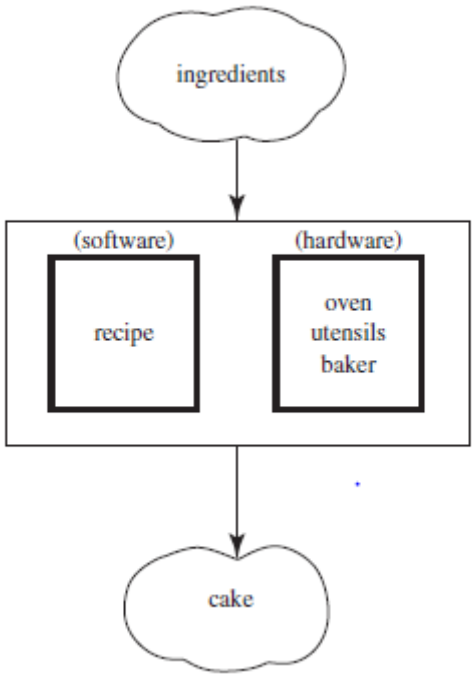


Komputery mogą różnić się wielkością (zgodnie z liczbą dostępnych bitów), rodzajami podstawowych operacji, które mogą wykonywać, szybkością wykonywania tych operacji, fizycznym nośnikiem zawierającym bity i ich wewnętrzną organizacją, oraz znacząco, w ich otoczeniu zewnętrznym. Ta ostatnia pozycja oznacza, że dwa komputery, które poza tym mają podobne funkcje, mogą wydawać się obserwatorowi bardzo różne: jeden może przypominać telewizor z klawiaturą, a drugi może być schowany pod pokrętłami i pokrętłami automatycznej dziewiarki. Jednak wygląd zewnętrzny ma znaczenie peryferyjne w porównaniu z bitami i ich wewnętrznym rozmieszczeniem. To właśnie bity „wyczuwają" zewnętrzne bodźce przychodzące ze świata zewnętrznego za pomocą przycisków, dźwigni, klawiszy na klawiaturze, elektronicznych linii komunikacyjnych, a nawet mikrofonów i kamer.

To właśnie bity „decydują” o tym, jak reagować na te bodźce i odpowiednio reagować, kierując inne bodźce na zewnątrz za pomocą wyświetlaczy, ekranów, drukarek, głośników, brzęczyków, dźwigni i korb. Jak robią to komputery? Co takiego przekształca tak trywialne operacje na bitach w niewiarygodne wyczyny, jakich dokonują komputery? Odpowiedź tkwi w głównych pojęciach : procesie i algorytmie, który go określa i powoduje, że ma miejsce.

Trochę gastronomii

Wyobraź sobie kuchnię, zawierającą zapas składników, szereg przyborów do pieczenia, piekarnik i (człowieka) piekarza. Pieczenie pysznego ciasta rodzynkowego to proces, który ze składników przeprowadza piekarz za pomocą piekarnika, a przede wszystkim zgodnie z przepisem. Składniki są danymi wejściowymi do procesu, ciasto jest jego wyjściem, a przepis jest algorytmem. Innymi słowy, algorytm określa czynności, które składają się na proces. Receptury lub algorytmy odnoszące się do zestawu omawianych procesów są ogólnie nazywane oprogramowaniem, podczas gdy przybory i piekarnik reprezentują to, co ogólnie nazywa się sprzętem. Piekarz w tym przypadku można uznać za część sprzętu.



Podobnie jak w przypadku operacji bitowych, konstelacja piekarz/piekarnik/przybory ma bardzo ograniczone możliwości bezpośrednie. Ten sprzęt do pieczenia ciast może nalewać, mieszać, rozsmarowywać, kapać, rozpalać piekarnik, otwierać drzwi piekarnika, mierzyć czas lub mierzyć ilości, ale nie może bezpośrednio piec ciast. To przepisy - te magiczne recepty, które zamieniają ograniczone możliwości sprzętu kuchennego w ciasta - a nie piekarniki czy piekarnie, są tematem tego tekstu. Przepisy, jak już wspomniano, nazywane są tutaj algorytmami, podczas gdy dziedzina badań nad ludźmi, wiedzy i ekspertyzy, która dotyczy algorytmów, będzie nazywana algorytmiką. Narysowana tu analogia została wykonana możliwie jak najdokładniej: przepis, który jest w pewnym sensie bytem abstrakcyjnym, jest algorytmem; formalna pisemna wersja przepisu, taka jak ta znaleziona w książce kucharskiej, jest analogiczna do programu komputerowego. Oprogramowanie w rzeczywistości odnosi się bardziej do programów — precyzyjnych reprezentacji algorytmów napisanych w specjalnych językach odczytywanych przez komputer — niż do samych algorytmów. Jednak dopóki nie omówimy

języków programowania, to rozróżnienie jest dość nieistotne. Konfrontujemy algorytmy, gdziekolwiek się udamy. Wiele codziennych procesów rządzi się algorytmami: wymiana przebitej opony, budowanie szafki zrób to sam, robienie na drutach swetra, dzielenie liczb, sprawdzanie numeru telefonu, aktualizowanie listy wydatków czy wypełnianie deklaracji podatkowej. Niektóre z nich (na przykład podział) mogą być w naszych umysłach bardziej bezpośrednio powiązane z komputerami, niż inne (na przykład budowa szaf), ale tutaj nas to dotyczy mniej. Chociaż komputery mają fundamentalne znaczenie dla tematu, w ogóle nie będziemy koncentrować się na ich fizycznych aspektach. To właśnie ich duchem zajmujemy się; z przepisami, które sprawiają, że działają zgodnie z ich algorytmami.

Algorytmika a informatyka

Algorytmika to coś więcej niż dziedzina informatyki. Jest to rdzeń informatyki i, uczciwie, można powiedzieć, że ma znaczenie dla większości nauki, biznesu i technologii. Sama natura algorytmiki sprawia, że jest ona szczególnie przydatna w tych dyscyplinach, które czerpią korzyści z wykorzystania komputerów, a te szybko stają się przytłaczającą większością. Wiadomo, że ludzie pytają: „Czym naprawdę jest informatyka? Dlaczego nie mamy nauki o łodziach podwodnych, nauki o zmywarkach ani nauki o telefonach?" Można argumentować, że telefony i zmywarki są tak samo ważne dla współczesnego życia jak komputery; może bardziej. Nieco bardziej skoncentrowanym pytaniem jest, czy informatyka obejmuje takie klasyczne dyscypliny, jak matematyka, fizyka, neurologia, elektrotechnika, językoznawstwo, logika i filozofia. My nie próbujemy odpowiedzieć na te pytania. Mamy jednak nadzieję, że domyślnie przekażemy coś z wyjątkowości i uniwersalności algorytmiki, a co za tym idzie, coś z wagi informatyki jako autonomicznej, choć młodej, dziedziny studiów. Ponieważ komputery mogłyby ograniczać ogólność algorytmiki, niektórzy uważają nieunikniony związek między nimi za niefortunny. W rzeczywistości określenie dziedziny „informatyka", jak powiedział ktoś kiedyś, jest jak odnoszenie się do chirurgii jako „nauki o nożach". Tak czy inaczej, jasne jest, że algorytmika nigdy nie rozwinęłaby się w taki sposób, jak bez tego łącza. Jednak ogólnie przyjmuje się, że termin „informatyka" jest mylący i że coś takiego jak „informatyka", „nauka o procesach" lub „nauka dyskretna" może być lepsze. Ponownie twierdzimy tylko, że nasza tematyka, algorytmika, stanowi podstawę informatyki, a nie ją zastępuje. Niektóre z tematów, które poruszamy w sequelu, takie jak istnienie problemów, których komputery nie mogą rozwiązać, mają implikacje filozoficzne, nie tylko na granicach wspaniałych maszyn, które jesteśmy w stanie zbudować, ale także na naszych własnych granicach jako śmiertelników o skończonej masie i skończonej żywotności. Pomimo głębokiej natury takich implikacji, w tej książce nacisk kładzie się na bardziej pragmatyczny cel, jakim jest zdobycie dogłębnego zrozumienia podstaw procesów wykonywanych maszynowo oraz receptur lub algorytmów, które nimi rządzą.

Trochę historii

Przyjrzyjmy się teraz kilku ważnym kamieniom milowym w rozwoju komputerów i informatyki, głównie po to, aby zilustrować, że jako uporządkowana dyscyplina naukowa jest to niezwykle młoda dziedzina. Gdzieś pomiędzy 400 a 300 rokiem p.n.e. wielki grecki matematyk Euklides wynalazł algorytm znajdowania największego wspólnego dzielnika (nwd) dwóch dodatnich liczb całkowitych. gcd X i Y jest największą liczbą całkowitą, która dokładnie dzieli X i Y . Na przykład gcd 80 i 32 wynosi 16. Szczegóły samego algorytmu nie mają tutaj znaczenia, ale algorytm Euklidesa, jak się go nazywa, jest uważany za pierwszy nietrywialny algorytm, jaki kiedykolwiek opracowano. Słowo algorytm wywodzi się od nazwiska perskiego matematyka Mohammeda al-Chuarizmi, który żył w IX wieku i któremu przypisuje się dostarczanie szczegółowych zasad dodawania, odejmowania, mnożenia i dzielenia zwykłych liczb dziesiętnych. Napisana po łacinie nazwa przekształciła się w Algorismus, od którego algorytm jest tylko małym krokiem. Najwyraźniej Euklides i al-Chuarizmi byli algorytmistami par excellence. Przechodząc od oprogramowania do sprzętu, jedną z pierwszych maszyn, które przeprowadzały proces

kontrolowany przez coś, co można by nazwać algorytmem, było krosno tkackie wynalezione w 1801 roku przez Francuza Josepha Jacquarda. Utkany wzór wyznaczały karty z dziurkami w różnych miejscach. Otwory te, wyczuwane przez specjalny mechanizm, sterowały doborem nici i innymi czynnościami maszyny. Interesujące jest to, że krosno Jacquard nie miało nic wspólnego z wąską konotacją numeryczną terminu „obliczenia". Jedną z najważniejszych i najbardziej barwnych postaci w historii informatyki był Charles Babbage. Ten angielski matematyk, po częściowym zbudowaniu w 1833 r. maszyny zwanej „silnikiem różnicowym" do oceny pewnych wzorów matematycznych, wymyślił i zaplanował niezwykłą maszynę, którą nazwał „silnikiem analitycznym". W przeciwieństwie do silnika różnicowego, który został zaprojektowany do realizacji określonego zadania, silnik analityczny miał być zdolny do wykonywania algorytmów lub programów, zakodowanych przez użytkownika jako dziurki w kartach. Gdyby skonstruowano silnik analityczny, byłby to matematyczny odpowiednik krosna Jacquarda, które w rzeczywistości było jego inspiracją. Nie trzeba dodawać, że maszyna Babbage'a była z natury mechaniczna, oparta na dźwigniach, zębatkach i przekładniach, a nie na elektronice i krzemie. Niemniej idee obecne w jego projekcie silnika analitycznego stanowią podstawę wewnętrznej struktury i działania współczesnych komputerów. Powszechnie uważa się, że Babbage żył na długo przed swoim czasem, a jego pomysły nie zostały docenione znacznie później. Ada Byron, hrabina Lovelace, była programistką Babbage'a. Jest jedną z najciekawszych postaci w historii informatyki, której przypisuje się położenie podwalin pod programowanie ponad sto lat przed pojawieniem się pierwszego działającego komputera. Amerykański inżynier Herman Hollerith wynalazł maszynę, również opartą na kartach dziurkowanych, która została wykorzystana przez Amerykańskie Biuro Spisu Ludności (American Census Bureau) do spisu ludności z 1890 roku. Jednak pierwsze komputery ogólnego przeznaczenia zbudowano dopiero w latach 40. XX wieku, częściowo jako odpowiedź na potrzeby obliczeniowe fizyków i astronomów, a częściowo jako naturalny wynik dostępności odpowiednich urządzeń elektromechanicznych i elektronicznych. Jak na ironię, pomogła też druga wojna światowa, z jej działalnością polegającą na budowaniu bomb i łamaniu kodów. Niektóre z kluczowych postaci w tym przełomowym i ekscytującym okresie to Anglik Alan Turing, Amerykanie Howard Aiken, John Mauchly, J. Presper Eckert i Herman Goldstine oraz słynny niemiecko-amerykański matematyk John von Neumann. Wracając do oprogramowania i algorytmiki, połowa lat 30. była świadkiem jednych z najbardziej fundamentalnych prac nad teorią algorytmów, przynoszących wyniki dotyczące możliwości i ograniczeń algorytmów wykonywanych maszynowo. Godne uwagi jest to, że praca ta, której fragmenty zostaną opisane w dalszej części , poprzedzała rzeczywistą materializację komputera. Niemniej jednak ma uniwersalne i trwałe znaczenie. Niektóre z kluczowych postaci, wszyscy matematycy, to znowu Alan Turing, Niemiec Kurt Gödel, Rosjanin Andrjej A. Markov i Amerykanie Alonzo Church, Emil Post i Stephen Kleene. Lata pięćdziesiąte i sześćdziesiąte były świadkami daleko idących i szybkich postępów technologicznych w projektowaniu i budowie komputerów. Można to przypisać z jednej strony nadejściem ery badań jądrowych i eksploracji kosmosu, a z drugiej boomowi dużych firm i banków oraz zróżnicowanej działalności rządowej. Precyzyjne przewidywanie różnych zjawisk jądrowych wymagało bardzo dużej mocy obliczeniowej, podobnie jak planowanie i symulacja misji kosmicznych. Eksploracja kosmosu wymagała również postępów w komunikacji wspomaganej komputerowo, ułatwiającej niezawodną analizę i filtrowanie, a nawet ulepszanie danych przesyłanych do i z satelitów i statków kosmicznych. Działalność biznesowa, bankowa i rządowa wymagała komputerów, które pomagałyby w przechowywaniu, manipulowaniu i wyszukiwaniu informacji dotyczących bardzo dużej liczby osób, pozycji inwentarzowych, danych fiskalnych i tak dalej. Interesujące dowody na znaczenie rozwoju technologicznego zorientowanego na maszyny w tym okresie można znaleźć w nazwach największej na świecie firmy komputerowej, IBM, i jednej z największych na świecie profesjonalnych organizacji związanych z komputerami, ACM. Pierwsza powstała około 1920 r., druga pod koniec lat 40. XX wieku. W obu przypadkach „M" pochodzi od słowa „machine": International Business Machines i Association for Computing Machinery. (IBM

wyewoluował z firmy utworzonej w 1896 r. przez wspomnianego Hermana Holleritha, aby produkować jego maszyny do tabulacji). Uznanie informatyki jako niezależnej dyscypliny akademickiej nastąpiło w połowie lat 60., kiedy kilka uniwersytetów utworzyło wydziały informatyki. W 1968 roku ACM opublikowała cieszącą się szerokim uznaniem rekomendację dotyczącą programu nauczania przedmiotów z informatyki, która stanowi podstawę większości aktualnych programów studiów z zakresu informatyki na poziomie licencjackim. Ten program jest okresowo aktualizowany. Dziś prawie każda instytucja akademicka posiada wydział informatyki lub grupę informatyczną w ramach wydziału matematyki lub elektrotechniki. Lata 60. pokazały ponowne zainteresowanie pracami z lat 30. nad algorytmiką i od tego czasu ta dziedzina jest przedmiotem szeroko zakrojonych i dalekosiężnych badań. Nie będziemy się więcej rozwodzić nad obecną sytuacją technologiczną: komputery są po prostu wszędzie. Używamy ich do surfowania po Internecie, co oznacza, że używamy ich do odbierania i dostarczania informacji, czytania, słuchania i oglądania oraz oczywiście przeglądania i kupowania. Istnieją komputery stacjonarne, laptopy i komputery wielkości dłoni, więc nigdy nie musimy ich bez nich obejść, a szybko zmniejszająca się przepaść między telefonami komórkowymi a komputerami zwiastuje erę komputerów do noszenia. Prawie każde nowoczesne urządzenie jest sterowane przez komputer, a na przykład jeden nowoczesny samochód zawiera ich dziesiątki. Dzieci proszą o komputery osobiste na urodziny i otrzymują je; studenci informatyki na większości uczelni muszą posiadać własne komputery do odrabiania prac domowych; i nie ma działalności przemysłowej, naukowej czy handlowej, która nie jest w sposób zasadniczy wspomagana przez komputery.

Dziwna dychotomia

Pomimo tego wszystkiego (lub prawdopodobnie w wyniku tego) opinia publiczna jest dziwnie podzielona, jeśli chodzi o umiejętność obsługi komputera. Wciąż są tacy, którzy nie wiedzą absolutnie nic o komputerach, a także członkowie stale rosnącej klasy informatyków. Poczynając od 10-letnich właścicieli komputerów osobistych, ta powiększająca się grupa osób korzystających na co dzień z komputerów obejmuje menedżerów, inżynierów, bankierów, techników i oczywiście profesjonalnych programistów, analityków systemowych i członków samego przemysłu komputerowego. Dlaczego to dziwne? Otóż, oto nauka, o której niektórzy ludzie nic nie wiedzą, ale o której szybko rosnąca liczba ludzi najwyraźniej wie wszystko! Tak się jednak składa, że naprawdę niezwykłym zjawiskiem jest to, że duże i ważne części informatyki nie są wystarczająco znane nie tylko członkom pierwszej grupy, ale także członkom drugiej grupy. Jednym z celów jest próba naświetlenia ważnego aspektu rewolucji komputerowej poprzez przedstawienie podstawowych pojęć, wyników i trendów leżących u podstaw nauki o obliczeniach. Skierowany jest zarówno do nowicjusza, jak i eksperta. Czytelnik bez wiedzy o komputerach (miejmy nadzieję) dowie się tutaj o ich „duchu” i sposobie myślenia, który zmusza ich do pracy, szukając gdzie indziej materiałów dotyczących ich „ciała”. Czytelnik znający się na komputerach, który może uznać, że pierwsze kilka rozdziałów jest raczej powolne, (mamy nadzieję) będzie mógł się wiele nauczyć od późniejszych.

Niektóre ograniczenia komputerów

Zanim rozpoczniemy naszą trasę, skontrastujmy pierwszy akapit tej części z niektórymi wyczynami, których obecne komputery nie są jeszcze w stanie wykonać. Powrócimy do tych kontrastów w ostatniej części, który dotyczy relacji między komputerami a ludzką inteligencją. Obecnie komputery są w stanie przeanalizować na miejscu ogromną ilość danych pochodzących z wielu zdjęć rentgenowskich mózgu pacjenta, zrobionych pod stopniowo rosnącymi kątami. Analizowane dane są następnie wykorzystywane przez komputer do wygenerowania przekrojowego obrazu mózgu, dostarczającego informacji o strukturze tkanek mózgu, umożliwiając w ten sposób precyzyjne zlokalizowanie takich nieprawidłowości jak guzy czy nadmiar płynów. W przeciwieństwie do tego, żaden obecnie dostępny komputer nie jest w stanie przeanalizować pojedynczego, zwykłego obrazu twarzy tego samego

pacjenta i określić jego wieku z marginesem błędu, powiedzmy, wynoszącym pięć lat. Jednak większość 12-letnich dzieci może! Jeszcze bardziej uderzająca jest zdolność rocznego dziecka do rozpoznawania twarzy matki na zdjęciu, którego nigdy wcześniej nie widziała, wyczyn, którego komputery nie są w stanie naśladować (i to nie tylko dlatego, że nie mają matek...) . Komputery są w stanie sterować w najbardziej precyzyjny i efektywny sposób, jaki można sobie wyobrazić, niezwykle wyrafinowanymi robotami przemysłowymi używanymi do konstruowania skomplikowanych części maszyn składających się z setek komponentów. W przeciwieństwie do tego, dzisiejsze najbardziej zaawansowane komputery nie są w stanie pokierować robotem, aby skonstruował ptasie gniazdo ze stosu gałązek, czego może dokonać każdy 12-miesięczny ptak! Dzisiejsze komputery potrafią grać w szachy na poziomie międzynarodowego arcymistrza, a tym samym mogą pokonać ogromną większość ludzkich graczy. Jednak przy bardzo nieznacznej zmianie zasad gry (na przykład poprzez umożliwienie skoczkowi dwóch ruchów naraz lub ograniczenie ruchów hetmana do pięciu pól), najlepsze z tych komputerów nie będą w stanie się przystosować bez przeprogramowania. lub zrekonstruowane przez ludzi. W przeciwieństwie do tego, 12-letni szachista amator będzie w stanie rozegrać całkiem dobrą partię z nowymi zasadami w bardzo krótkim czasie, a wraz z doświadczeniem stanie się coraz lepszy. Jak wspomniano, te różnice są związane z różnicą między inteligencją ludzką a skomputeryzowaną. Będziemy w lepszej pozycji do dalszego omówienia tych kwestii w rozdziale 15, po tym, jak dowiemy się więcej o algorytmach i ich właściwościach.

Przepis

Oto przepis na mus czekoladowy zaczerpnięty z francuskiej kuchni Sinclaira i Malinowskiego. Składniki - to znaczy wkłady - obejmują 8 uncji półsłodkich kawałków czekolady, 2 łyżki wody, 14 filiżanek cukru pudru, 6 oddzielnych jajek i tak dalej. Daje od sześciu do ośmiu porcji pysznego musu z czekoladą. Oto przepis lub algorytm. Rozpuść czekoladę i 2 łyżki wody w podwójnym bojlerze. Po roztopieniu dodaj cukier puder; dodawać po trochu masło. Odłożyć na bok. Ubijaj żółtka do gęstej i cytrynowej barwy, około 5 minut. Delikatnie zawinąć w czekoladę. W razie potrzeby lekko podgrzej, aby rozpuścić czekoladę. Dodaj rum i wanilię. Białka ubić do uzyskania piany. Ubij 2 łyżki cukru; ubijaj, aż uformują się sztywne szczyty. Delikatnie złóż białka w mieszankę czekoladowo-żółtkową. Wlać do poszczególnych porcji dań. Schłodź co najmniej 4 godziny. W razie potrzeby podawać z bitą śmietaną. Robi od 6 do 8 porcji.

Jest to „oprogramowanie" związane z przygotowaniem musu; jest to algorytm, który kontroluje proces wytwarzania musu ze składników. Sam proces jest wykonywany przez „sprzęt", w tym przypadku osobę przygotowującą mus, wraz z różnymi przyborami: podwójnym bojlerem, urządzeniem grzewczym, trzepaczką, łyżkami, minutnikiem i tak dalej.

Poziomy szczegółowości

Przyjrzyjmy się bliżej najbardziej elementarnym instrukcjom zawartym w tym przepisie. Rozważ instrukcję „domieszaj cukier puder". Dlaczego przepis nie mówi „weź trochę cukru pudru, wlej go do roztopionej czekolady, wmieszaj, weź trochę więcej, wlej, wymieszaj, . . .? A dokładniej, dlaczego nie jest napisane „weź 2365 ziaren cukru pudru, wlej je do roztopionej czekolady, weź łyżkę i mieszaj okrężnymi ruchami, . . .? Lub, by być jeszcze bardziej precyzyjnym, dlaczego nie „przesunąć ręką w kierunku składników pod kątem 14◦, z przybliżoną prędkością 18 cali na sekundę, . . .? Odpowiedź jest oczywiście oczywista. Sprzęt wie, jak wymieszać cukier puder z rozpuszczoną czekoladą i nie wymaga dalszych szczegółów. A co powiesz na odwrócenie sprawy i pytanie, czy to możliwe, że sprzęt wie, jak przygotować mieszankę czekolady z cukrem i masłem? W takim przypadku całą pierwszą część przepisu można by zastąpić prostą instrukcją „przygotuj mieszankę czekoladową". Doprowadzając to do skrajności, może sprzęt wie, jak przygotować mus czekoladowy. Umożliwiłoby to zastąpienie całego

przepisu „przygotuj mus czekoladowy”. Przy takim poziomie wiedzy o sprzęcie, pojedyncza linijka instrukcji jest idealnym przepisem na uzyskanie musu z czekolady; ten krótki przepis jest jasny, nie zawiera błędów i gwarantuje uzyskanie pożądanych wyników. Takie eksperymenty myślowe jasno pokazują, że poziom szczegółowości jest bardzo ważny, jeśli chodzi o podstawowe instrukcje algorytmu. Musi być dostosowany do konkretnych możliwości sprzętu, a także powinien być dostosowany do poziomu zrozumienia potencjalnego czytelnika lub użytkownika algorytmu. Rozważmy inny przykład poznany na początku naszego życia, który jest nieco bliższy obliczeniom - uporządkowane mnożenie liczb. Załóżmy, że zostaniemy poproszeni o pomnożenie 528 przez 46. Dokładnie wiemy, co robić. Pomnożymy 8 przez 6, otrzymując 48, zapisujemy cyfrę jednostki wyniku 8 i pamiętamy cyfrę dziesiątek 4; następnie mnożymy 2 przez 6 i dodajemy 4, otrzymując 16; zapisujemy cyfrę jednostek 6 po lewej stronie 8 i pamiętamy cyfrę dziesiątek 1; i tak dalej. Tutaj można zadać te same pytania. Dlaczego „mnożyć 8 przez 6?” Dlaczego nie „sprawdzić wpisu znajdującego się w ósmym wierszu i szóstej kolumnie tabliczki mnożenia” lub „dodać 6 do siebie 8 razy”? Podobnie, dlaczego nie możemy rozwiązać całego problemu za jednym pociągnięciem za pomocą prostego i satysfakcjonującego algorytmu „pomnóż dwie liczby?” To ostatnie pytanie jest dość subtelne: dlaczego możemy bezpośrednio pomnożyć 8 przez 6, ale nie 528 przez 46? Ponownie, jasne jest, że poziom szczegółowości jest kluczową cechą naszej akceptacji algorytmu mnożenia. Zakładamy, że odpowiedni sprzęt (w tym przypadku my sami) jest w stanie wykonać 8 razy 6, ale nie 528 razy 46, i że możemy to zrobić w naszych głowach, a przynajmniej znamy jakiś inny sposób, aby nie trzeba było nam mówić, jak sprawdzić wynik w tabeli. Te przykłady pokazują potrzebę uzgodnienia na samym początku podstawowych działań, które algorytm uważa za zdolny do przepisania. Bez tego nie ma sensu szukać algorytmów do czegokolwiek. Co więcej, różne problemy są naturalnie związane z różnymi rodzajami podstawowych działań. Przepisy wymagają mieszania, mieszania, nalewania i podgrzewania; mnożenie liczb pociąga za sobą dodawanie, mnożenie cyfr i, co ważne, zapamiętywanie cyfry; wyszukanie numeru telefonu może wymagać odwrócenia strony, przesunięcia palcem w dół listy i porównania podanego nazwiska z tym, na które wskazujemy. W precyzyjnych rodzajach algorytmów, które będziemy omawiać, te podstawowe instrukcje muszą być sformułowane jasno i precyzyjnie. Nie możemy zaakceptować rzeczy takich jak „ubijanie białek do uzyskania piany”, ponieważ wyobrażenie jednej osoby na pianę może być zupełnie niepodobne do innej! Instrukcje muszą być odpowiednio odróżnialne od nieinstrukcyjnych, takich jak „robi 6 do 8 porcji”. Zwroty rozmyte, takie jak „około 5 minut”, nie mają miejsca w algorytmie przystosowanym do wykonywania przez komputer, jak ma to miejsce w przypadku niejasności, takich jak „podawaj z bitą śmietaną, jeśli chcesz”. (Czy to faktyczna porcja, czy dodanie bitej śmietany, zależy od pragnień danej osoby?) Przepisy na mus, w przeciwieństwie do algorytmów, które będą nas interesować, zbyt wiele rzeczy uznają za oczywiste, z których najważniejsza czyli fakt, że człowiek jest częścią sprzętu. Nie możemy polegać na tego rodzaju luksusie, dlatego musimy być znacznie bardziej wymagający. Ogólna jakość algorytmu zależy przede wszystkim od wyboru dozwolonych działań podstawowych i ich adekwatności do danej sprawy.

Abstrakcja

Wcześniej stwierdzono, że prawdziwe komputery mogą wykonywać tylko niezwykle proste operacje na niezwykle prostych obiektach. Może się to wydawać sprzeczne z obecną dyskusją, w której zaleca się projektowanie różnych algorytmów przy użyciu podstawowych działań o różnym poziomie szczegółowości. Jednak analogia jest nadal aktualna. Uczniowi szefa kuchni może być konieczne podanie przepisu na mus czekoladowy, ale po kilku latach przygotowania musu wystarczy instrukcja „przygotuj mus czekoladowy”. Mówimy, że pojęcia takie jak „mus czekoladowy”, „beza cytrynowa” i „krem bawarski” są na wyższym poziomie abstrakcji niż operacje takie jak „mieszanie”, „mieszanie” i „przelewanie” stosowane w przepisach na ich przygotowanie. W ten sam sposób, dzięki odpowiedniemu programowaniu, komputer może rozpoznawać abstrakcje wyższego poziomu, takie

jak liczby, tekst i obrazy. Podobnie jak w gotowaniu, w komputerze istnieje wiele poziomów abstrakcji, z których każdy jest odpowiedni do opisywania różnych rodzajów algorytmów. Na przykład ten sam komputer jest postrzegany inaczej przez 12-latka grającego w grę komputerową, przez jego siostrę, która surfuje po Internecie, przez ojca, który używa arkusza kalkulacyjnego do obliczania ocen uczniów, a przez matkę. kto pisze program zarządzania testem skuteczności nowej szczepionki. Żadne z nich nie zna ani nawet nie dba o bity, które naprawdę składają się na proces obliczeniowy, którego używają. Ten proces abstrahowania od szczegółów w celu dostrzeżenia wspólnych wzorców w pozostałej części jest sednem prawie każdego ludzkiego przedsięwzięcia. Na przykład czytanie książki ma wpływ na mózg, który składa się z kilku odrębnych regionów, z których każdy składa się z neuronów i innych komórek. Komórki te zbudowane są ze złożonych cząsteczek, które zbudowane są z atomów, które z kolei zbudowane są z bardziej elementarnych cząstek. Wszystkie te różne poziomy abstrakcji odnoszą się do tego, co dzieje się w twoim mózgu, ale nie można ich wszystkich rozpatrywać łącznie. W rzeczywistości należą one do różnych dziedzin nauki: fizyki cząstek elementarnych, chemii, biologii molekularnej, neurobiologii i psychologii. Psycholog przeprowadzający eksperymenty z krótkotrwałą retencją pamięci będzie rozpraszany tylko przez myślenie o związkach między atomami i cząsteczkami w mózgu. To samo dotyczy informatyki. Gdybyśmy byli zmuszeni do myślenia na poziomie bitowym przez cały czas, komputer nie byłby przydatny. Zamiast tego możemy na przykład pomyśleć o grupie bitów (zwykle osiem bitów lub „bajt") jako oznaczającej znak. Możemy teraz rozważyć sekwencje bajtów do oznaczania słów angielskich, sekwencje słów i znaki interpunkcyjne do oznaczania zdań itd. do akapitów, rozdziałów i książek. Dla każdego z tych poziomów istnieją algorytmy. Na przykład sprawdzanie pisowni dotyczy słów, ale nie znaków, justowanie do lewej dotyczy akapitów, a tworzenie spisu treści dotyczy książek. W każdym przypadku możemy opisać algorytm, całkowicie ignorując fragmenty składające się na słowa, akapity lub całe książki. W miarę rozwoju, będziemy omawiać środki techniczne, które pozwalają nam tworzyć takie abstrakcje. Tymczasem opiszemy każdy algorytm na odpowiednim dla niego poziomie abstrakcji.

Krótkie algorytmy dla długich procesów

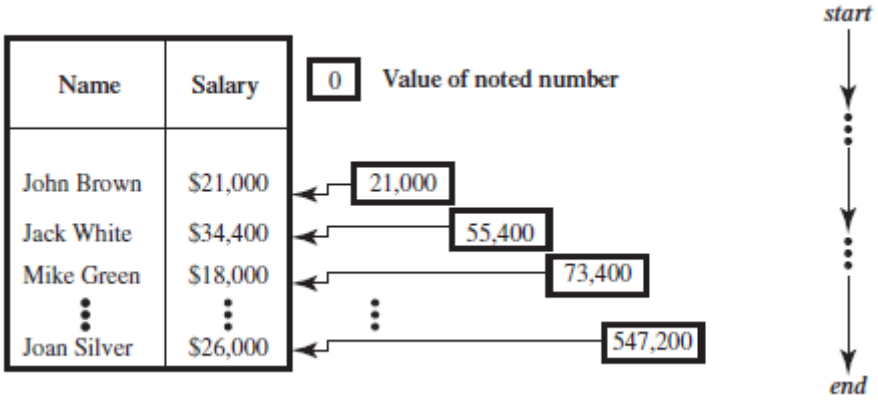
Załóżmy, że otrzymujemy listę akt personalnych, po jednym dla każdego pracownika w określonej firmie, z których każda zawiera imię i nazwisko pracownika, dane osobowe oraz wynagrodzenie. Interesuje nas łączna suma wszystkich wynagrodzeń wszystkich pracowników. Oto algorytm wykonania tego zadania:

(1) zanotuj cyfrę 0;

(2) przejdź przez listę, dodając wynagrodzenie każdego pracownika do zanotowanej liczby;

(3) po dojściu do końca listy, wygeneruj odnotowaną liczbę jako wynik.

Zanim przejdziemy dalej, powinniśmy najpierw przekonać się, że ten prosty algorytm spełnia swoje zadanie. „Zanotowana" liczba, którą można uznać za zapamiętaną lub zapisaną na kartce papieru, zaczyna się od wartości zero. Po dokonaniu doliczenia w ust. 2 dla pierwszego pracownika, liczba ta faktycznie przyjmuje wartość wynagrodzenia tego pracownika. Po drugim pracowniku jego wartość jest sumą wynagrodzeń pierwszych dwóch pracowników. Ostatecznie jego wartość jest wyraźnie sumą wszystkich wynagrodzeń.

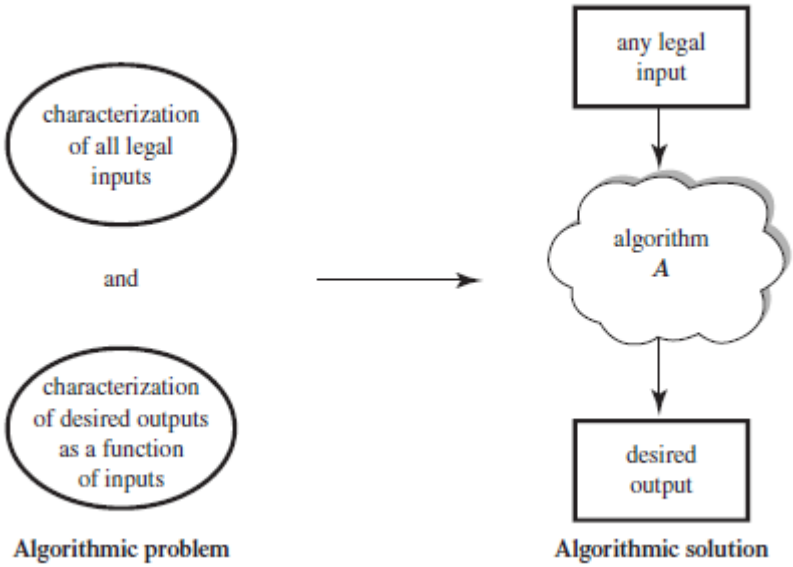


Interesujące jest to, że tekst tego algorytmu jest krótki i ma stałą długość, ale proces, który opisuje i kontroluje, zmienia się wraz z długością listy pracowników i może być bardzo, bardzo długi. Dwie firmy, pierwsza z jednym pracownikiem, a druga z milionem, mogą wprowadzić swoją listę pracowników do tego samego algorytmu, a problem sumowania wynagrodzeń zostanie rozwiązany dla każdej z nich równie dobrze. Oczywiście w przypadku pierwszej firmy proces ten nie potrwa długo, natomiast w przypadku drugiej będzie to dość długi czas. Algorytm jest jednak ustalony. Nie tylko tekst algorytmu jest krótki i ma stałą wielkość, ale zarówno mała jak i duża firma potrzebuje tylko jednej zanotowanej liczby, aby wykonać zadanie, dzięki czemu ilość „przyborów" jest tutaj również mała i stała. Oczywiście potencjalna wartość podanej liczby będzie prawdopodobnie musiała być większa dla większych firm, ale cały czas będzie tylko jedna liczba

Problem algorytmiczny

I tak mamy ustalony algorytm określający wiele procesów o różnej długości, dokładny czas trwania i charakter procesu w zależności od danych wejściowych do algorytmu. Rzeczywiście, nawet prosty przykład sumowania wynagrodzeń pokazuje różne możliwe dane wejściowe: firmy jednoosobowe, firmy liczące milion osób, firmy, w których część pensji wynosi zero lub te, w których wszystkie pensje są równe. Czasami algorytm musi również działać z dziwacznymi danymi wejściowymi, takimi jak firmy bez pracowników lub takie, które zatrudniają ludzi otrzymujących ujemne pensje (czyli pracowników, którzy płacą firmie za przyjemność pracy dla niej). W rzeczywistości algorytm wynagrodzeń powinien działać zadowalająco dla nieskończonej liczby wejść. Istnieje nieskończona liczba całkowicie akceptowalnych list pracowników, a algorytm powinien być w stanie zsumować pensje w dowolnej z nich, gdy zostanie podany jako dane wejściowe. Ta kwestia nieskończenie wielu potencjalnych danych wejściowych nie do końca pasuje do analogii przepisu, ponieważ chociaż przepis powinien działać doskonale bez względu na to, ile razy jest używany, jego składniki są zwykle opisywane jako ustalone w ilości, a zatem w istocie przepis ma tylko jeden potencjalny wkład (przynajmniej w ilościach; oczywiście cząsteczki i atomy będą za każdym razem inne). Jednak przepis na mus czekoladowy mógł być ogólny; to znaczy, jego lista składników mogła brzmieć coś w stylu „X uncji kawałków czekolady, X/4 łyżek wody, X/32 filiżanki cukru pudru itp.", a jego ostatnia linia mogła brzmieć „sprawia, że 3X/4 do X porcji." Byłoby to bardziej zgodne z prawdziwym pojęciem algorytmu. W obecnej formie przepis jest algorytmem o nieco banalnym charakterze, ponieważ jest dostosowany do jednego konkretnego zestawu składników. Może być przeprowadzana (lub, w terminologii algorytmicznej, może być uruchamiana lub wykonywana) kilka razy, ale z zasadniczo tymi samymi danymi wejściowymi, ponieważ jedna filiżanka mąki jest uważana za dokładnie taką samą jak każda inna. Same dane wejściowe muszą być zgodne z przeznaczeniem algorytmu. Oznacza to na przykład, że lista bestsellerów New York Times nie byłaby akceptowana jako dane wejściowe do algorytmu sumowania

wynagrodzeń, podobnie jak masło orzechowe i galaretka nie byłyby akceptowane jako składniki przepisu na mus. Pociąga to za sobą pewien rodzaj specyfikacji dozwolonych wejść. Ktoś musi dokładnie określić, które listy pracowników są legalne, a które nie; gdzie dokładnie na liście znajduje się wynagrodzenie; czy jest podana odręcznie (na przykład 32 000 USD), czy może w jakiejś skróconej formie (na przykład 32 000 USD); gdzie kończy się rekord pracownika, a zaczyna kolejny i tak dalej. Mówiąc najogólniej, przepisy lub algorytmy są rozwiązaniami na pewno rodzaje problemów, zwanych problemami obliczeniowymi lub algorytmicznymi. W przykładzie płacowym problem można sprecyzować w postaci prośby o numer reprezentujący sumę wynagrodzeń z listy pracowników organizacji. Lista ta może mieć różną długość, ale musi być ułożona w określony sposób. Problem taki można postrzegać jako poszukiwanie zawartości „czarnej skrzynki", która jest określona przez precyzyjną definicję legalnych danych wejściowych i precyzyjną definicję wymaganych wyników jako funkcji tych danych wejściowych; czyli sposób, w jaki każde wyjście zależy od wejścia.



Problem algorytmiczny został rozwiązany po znalezieniu odpowiedniego algorytmu. Czarna skrzynka została wtedy faktycznie zaopatrzona w zawartość; to „działa" zgodnie z tym algorytmem. Innymi słowy, czarna skrzynka może wytworzyć odpowiednie dane wyjściowe z dowolnego legalnego wkładu, wykonując proces, który jest określony i zarządzany przez ten algorytm. Słowo „dowolny" w poprzednim zdaniu jest bardzo ważne. Nie interesują nas rozwiązania, które nie działają dla wszystkich podanych wejść. Łatwo jest znaleźć rozwiązanie, które działa dobrze tylko w przypadku niektórych legalnych danych wejściowych. Jako skrajny przykład trywialny algorytm:

(1) wyprodukuj 0 jako wyjście.

bardzo dobrze sprawdza się w przypadku kilku interesujących list pracowników: tych bez pracowników, tych, w których każdy zarabia 0,00 zł (lub ich wielokrotności), a także tych z listą płac, która odzwierciedla idealną równowagę między pensją dodatnią a ujemną. Później zajmiemy się takimi zagadnieniami, jak wydajność i praktyczność algorytmów. Tutaj stawiamy minimalny wymóg, że algorytm faktycznie rozwiązuje problem, nawet jeśli może to zrobić nieefektywnie. Oczywiście sam problem może określać wymagane zachowanie potencjalnego algorytmu na niepożądanych danych wejściowych, ale wtedy te dane wejściowe, chociaż niepożądane, są nadal legalne. Na przykład problem sumowania wynagrodzeń mógłby zawierać wymóg, aby w przypadku pracownika, którego

rekord nie zawierał numeru w obszarze wynagrodzeń, ale powiedzmy znak zapytania lub inne bezsensowne dane, algorytm powinien dodać nazwisko tego pracownika do specjalnej listy, która zostanie przekazana do biura płac do dalszych działań. Taka niekonwencjonalna lista pracowników jest jednak legalna; po prostu nie jest traktowana w standardowy sposób, ale jest traktowana w specjalny sposób, który pasuje do jego nienormalnej natury. W związku z tym trzymanie nielegalnych danych wejściowych oddzielnie jest odpowiedzialnością problemu algorytmicznego, podczas gdy traktowanie specjalnych klas nietypowych lub niepożądanych danych wejściowych jest odpowiedzialnością samego algorytmu.

Granice podstawowych działań

Jest jeszcze jedna ważna sprawa, którą musimy się w tym miejscu poruszyć, dotycząca wykonania podstawowych czynności lub operacji, zaleconych przez algorytm. Oczywistym jest, że każda z tych czynności musi być wykonana w skończonym czasie, inaczej oczywiście algorytm nigdy się nie skończy. Tak więc nieskończenie długie działania są złe. Działania, które mogą zająć nieskończenie małą ilość czasu, są również zakazane, co nie wymaga uzasadnienia. Jest nie do pomyślenia, aby maszyna kiedykolwiek była w stanie wykonywać czynności w coraz krótszym czasie. Na przykład prędkość światła zawsze byłaby ograniczeniem prędkości każdej maszyny. Podobne ograniczenia zasobów (czyli narzędzi) wykorzystywanych do wykonywania podstawowych czynności również muszą być narzucone, ale nie będziemy tu omawiać przyczyn. Jasne jest, że te założenia dotyczące podstawowych działań rzeczywiście obowiązują w przypadku prawdziwych komputerów. Na przykład podstawowe akcje manipulacji bitami są precyzyjne i jednoznaczne oraz zajmują ograniczoną ilość czasu i zasobów. Tak więc, zgodnie z obietnicą, opisana tu teoria algorytmiki będzie miała bezpośrednie zastosowanie do problemów przeznaczonych do rozwiązania komputerowego.

Problem i jego rozwiązanie: podsumowanie

Podsumowując, problem algorytmiczny składa się z:

1. scharakteryzowanego legalnego, możliwie nieskończonego zbioru potencjalnych zbiorów wejściowych, oraz

2. specyfikację pożądanych wyjść jako funkcję wejść.

Zakłada się, że z góry podany jest również opis dozwolonych akcji podstawowych lub konfiguracja sprzętowa wraz z jej wbudowanymi akcjami podstawowymi. Rozwiązaniem problemu algorytmicznego jest algorytm składający się z elementarnych instrukcji zalecających działania z uzgodnionego zbioru. Algorytm ten, gdy jest wykonywany dla dowolnego dozwolonego zestawu danych wejściowych, rozwiązuje problem, wytwarzając wymagane dane wyjściowe.

Ważne jest, aby rozpoznać znaczne trudności związane z zadowalającym rozwiązywaniem problemów algorytmicznych. Zaczynając od przepisu na mus, a następnie podając prosty algorytm sumowania, popełniono pewną niesprawiedliwość, ponieważ mogłoby się wydawać, że sprawy są łatwe. Nic nie jest dalsze od prawdy. W praktyce problemy algorytmiczne mogą być niezwykle złożone, a ich pomyślne rozwiązanie może zająć lata pracy. Co gorsza, jak zobaczymy dalej, wiele problemów nie daje zadowalających rozwiązań, podczas gdy inne w ogóle nie dopuszczają żadnych rozwiązań. W przypadku wielu problemów status, jeśli chodzi o dobre rozwiązania algorytmiczne, jest jeszcze nieznany, mimo intensywnej pracy wielu utalentowanych ludzi. Oczywiście nie będziemy w stanie zilustrować zagadnień poruszanych tu zbyt obszernymi i złożonymi przykładami, ale możemy wyczuć trudność w projektowaniu algorytmów, myśląc o następujących (nieformalnie opisanych) problemach algorytmicznych. W pierwszym zadaniu dane wejściowe to legalna pozycja w szachach (czyli opis

sytuacji osiągniętej w pewnym momencie podczas partii szachów), podczas gdy dane wyjściowe to najlepszy ruch dla białych (czyli opis ruchu, który maksymalizuje szanse białych na wygranie meczu). Drugi problem dotyczy dystrybucji gazet. Załóżmy, że 20 000 gazet ma zostać rozprowadzonych do 1000 lokalizacji w 100 miastach przy użyciu 50 ciężarówek. Dane wejściowe zawierają odległości drogowe między miastami i lokalizacjami w każdym mieście, liczbę dokumentów wymaganych w każdej lokalizacji, obecną lokalizację każdej ciężarówki, zdolność każdej ciężarówki do przewozu gazet, a także pojemność benzyny i przebieg -wydajność galonów i szczegóły dotyczące dostępnych kierowców, w tym ich aktualne miejsce pobytu. Wynikiem ma być lista, dopasowująca kierowców do ciężarówek i zawierająca szczegółowe trasy dla każdej z ciężarówek, aby zminimalizować całkowitą liczbę przejechanych mil. W rzeczywistości problem wymaga algorytmu, który działa dla dowolnej liczby gazet, lokalizacji, miast i ciężarówek, tak aby ich liczba również się różniła i stanowiła część danych wejściowych. Zanim będziemy mogli omówić kwestie poprawności i skuteczności, czy też głębsze pytania dotyczące natury lub samego istnienia rozwiązań pewnych problemów algorytmicznych, musimy dowiedzieć się więcej o budowie algorytmów i strukturze obiektów, którymi manipulują.