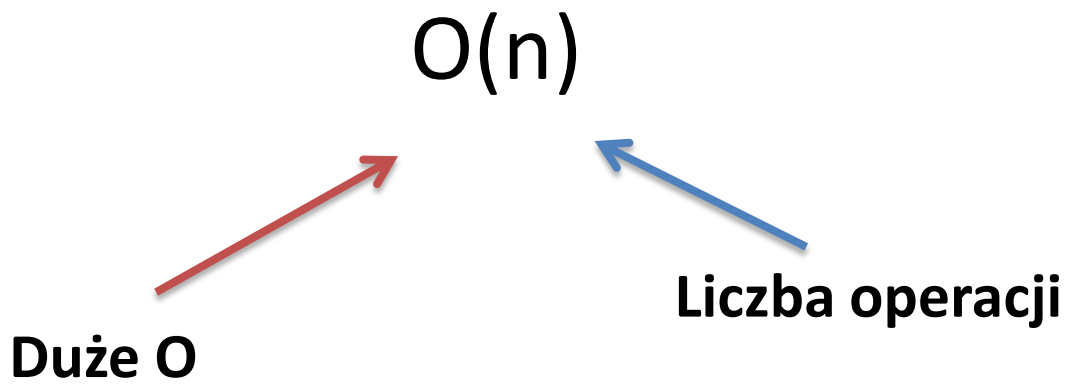
Złożoność obliczeniowa
Notacja dużego O.

W wyborze lub w tworzeniu algorytmów istotne jest to, aby algorytm zajmował jak najmniej pamięci (miał jak najkrótszy kod) oraz, żeby był w krótkim czasie wykonywany. Mimo, że pamięci komputerów są coraz większa i są coraz szybsze procesory, to wiele problemów informatycznych wciąż pozostaje bez rozwiązania.

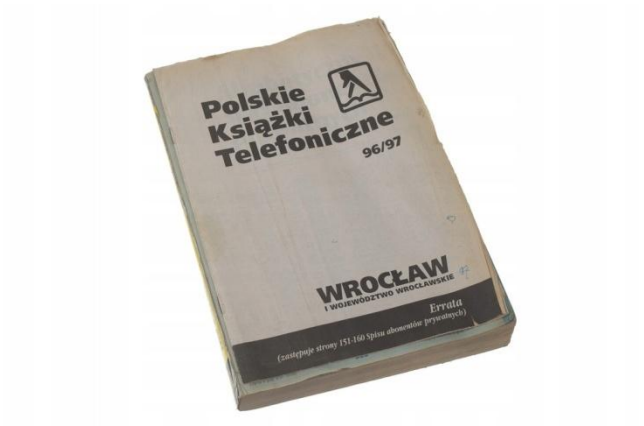
Czas wykonywania algorytmu jest bardzo ważnym zagadnieniem w informatyce. W niemal każdym zadaniu brana jest pod uwagę tzw. notacja dużego O , czyli opis szybkości algorytmu. Poniżej notacja dużego O.



Złożoność obliczeniowa
Notacja dużego O.

Notacja dużego O, to oszacowanie złożoności obliczeniowej algorytmu, czyli przede wszystkim ile operacji wykona dany algorytm.
Na przykład mamy listę (tablicę) o rozmiarze n . Wyszukiwanie proste sprawdzi każdy element po kolei, więc wykona n operacji. W notacji dużego O czas wykonywania algorytmu wyrazimy zapisem O(n). Taki zapis nazywamy także funkcją. Litera w nawiasie jest argumentem funkcji – liczbą operacji jakiegoś działania w kodzie. Czas wykonania będzie zależał od długości listy, ale złożoność obliczeniowa tego algorytmu to O(n).

Wyszukiwanie proste O(n)



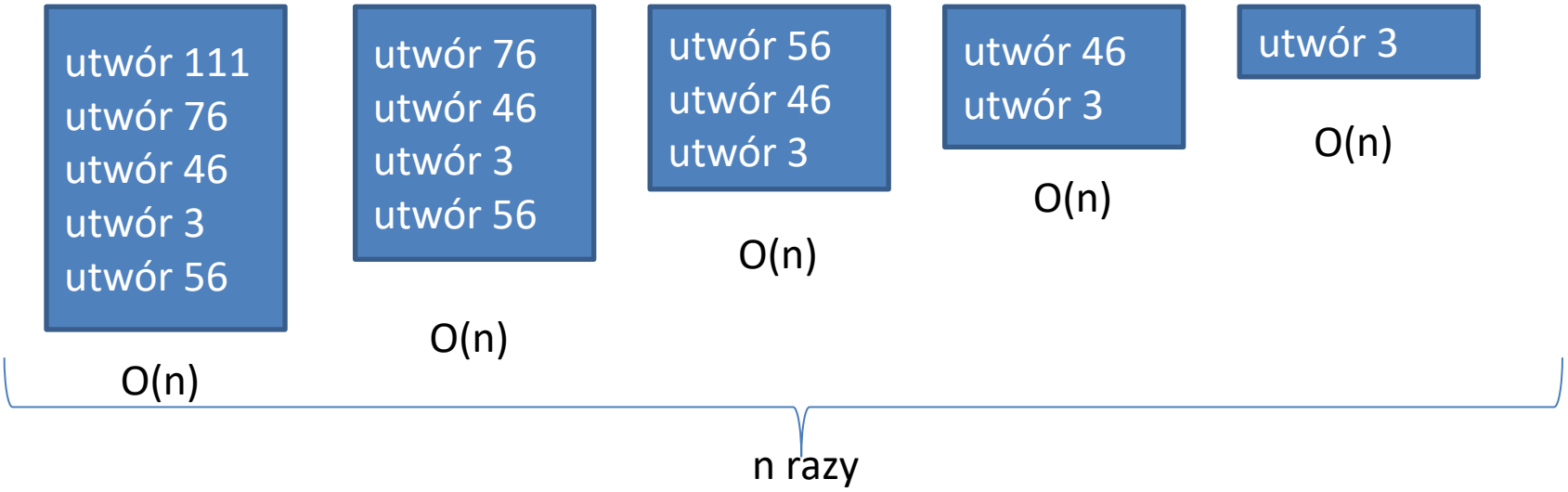
Dobrym przykładem wykonania algorytmu o złożoności obliczeniowej O(n) będzie wyszukiwanie nazwiska w książce telefonicznej. W najlepszym przypadku możemy odnaleźć szukanego nazwiska na pierwszym miejscu spisu (listy, tablicy).

Złożoność obliczeniowa

Złożoność może wynieść nawet O(1). W najgorszym przypadku przejrzymy wszystkie pozycje w książce, których jest np. 500 000, czyli O(500 000). Algorytm wyszukiwania prostego będzie wyrażony w notacji O(n) niezależnie jak długa jest lista do wyszukiwania. Notacja dużego O opisuje najgorszy przypadek.

Sortowanie przez wybór $O(n^2)$

Istnieją różne złożoności i jej zapisy w postaci notacji dużego O.
Zapis **$O(n^2)$** mówi, że ilość operacji algorytmu, będzie więcej i dłuższy będzie czas. Algorytm o takiej złożoności to np. sortowanie przez wybieranie.
Przykładem może być wybieranie z listy utworów najczęściej słuchanych od największej do najmniejszej. Musimy przeszukiwać listę tyle razy ile jest utworów i na innej liście układać je w kolejności malejącej.



Złożoność obliczeniowa
Notacja dużego O.

utwór 111
utwór 76
utwór 56
utwór 46
utwór 3

Posortowana lista . Czas wykonania wynosi O(n*n) czyli $O(n^2)$.

Wyszukiwanie binarne O(log n)

Wyszukiwanie binarne (połówkowe, przez połowienie) jest algorytmem o złożoności O(log n). Wyszukiwanie binarne przeszukuje listę uporządkowaną (posortowaną) i w przeciwieństwie do wyszukiwania prostego nie sprawdza każdego elementu listy, tylko dzieli ją na pół i sprawdza , w którejś z połówek. Potem znów dzieli na pół … Podobnie jak

Złożoność obliczeniowa
Notacja dużego O.

w zabawie zgadnij liczbę. Podajemy zgadującemu zakres liczb, wymyślamy jakąś liczbę i sprawdzamy odpowiedzi zgadującego słowami większa lub mniejsza. Jeśli zgadujący zna algorytm wyszukiwania binarnego, to szybko poradzi sobie z problemem, w przeciwnym razie trochę sobie postrzela.

Z posortowanej listy liczb od 1 do 100 szukamy 1.
1) Dzielimy 100/2 =50 i sprawdzamy czy to ta liczba. Okazuje się, że jest mniejsza.
2) Dzielimy 50/2 =25 znów mniejsza
3) Dzielimy 25/2 = 13 (zaokrąglamy do większej) dalej mniejsza
4) Dzielimy 13/2 = 7 mniejsza
5) Dzielimy 7/2= 4 mniejsza
6) Dzielimy 4/2= 2 mniejsza
7) Dzielimy 2/2 = 1 jest
Szukaną liczbę znaleźliśmy po 7 próbach.
Wyszukiwanie binarne w każdej próbie eliminuje połowę szukanych liczb.

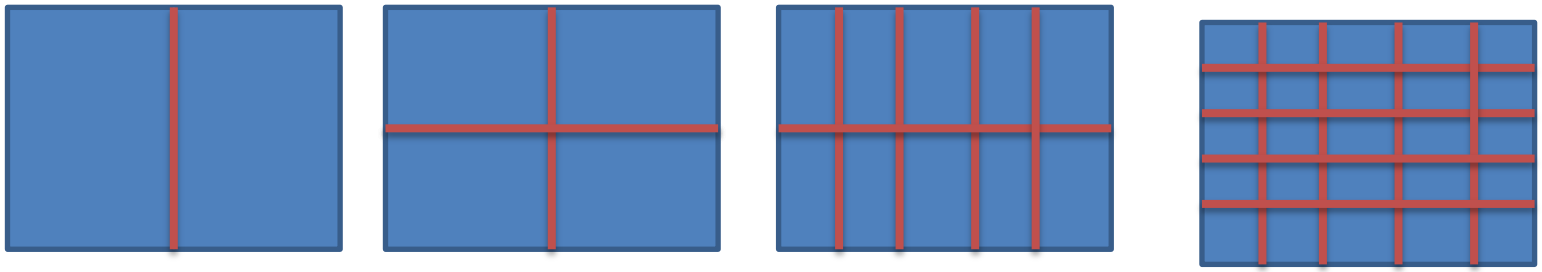
Złożoność obliczeniowa
Notacja dużego O.

Co nam mówi zapis O(log n). Małe n to ilość elementów do przeszukania. Zapis log to w matematyce odwrotność potęgowania. Potęgowanie wygląda tak: $2^3 = 8$. Logarytm (log) zapyta : do jakiej potęgi należy podnieść liczbę 2, aby otrzymać 8. Odpowiedź: 3. Tzn., że log 8 = 3. Ten zapis logarytmu (bez podanej podstawy np. $\log_{10}$) mówi nam , że ma podstawę 2 ($\log_2$). Czyli log = $\log_2$.

Podstawa logarytmu

Mamy do wyszukania liczbę 30 w 32 elementowej , **posortowane**j liście. Jaki algorytm będzie szybszy O(n) czy O(log n). Pierwszy wykona co najmniej 30 operacji, drugi log 32 = 5. Pięć operacji.

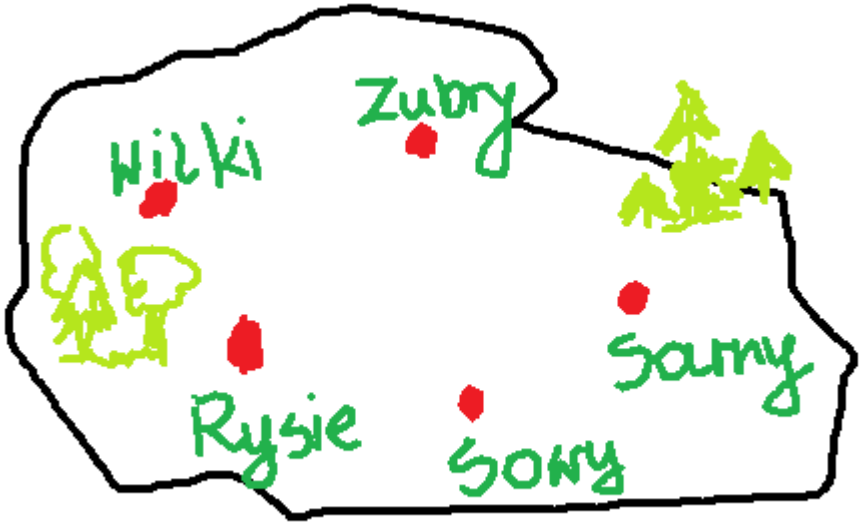
Co zrobimy szybciej? Narysujemy 16 prostokątów na kartce do ksero, czy zegniemy ją na 4 części? Najpierw zginamy na pół, potem znowu na pół … i już , log 16 = 4 , zamiast rysowania 16 prostokątów.



Złożoność obliczeniowa
Notacja dużego O.

Problem komiwojażera O(n!)

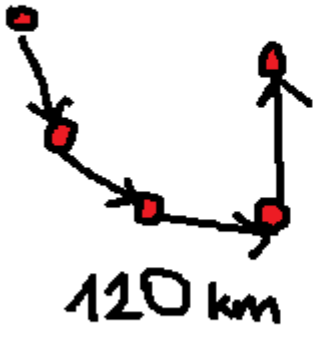
Algorytm ten charakteryzuje się bardzo słabym czasem wykonania. W informatyce problem ten jest powszechnie znany, ponieważ cechuje go wyjątkowe tempo wzrostu złożoności. Znany jest też pod nazwą problemem podróżującego komiwojażera.

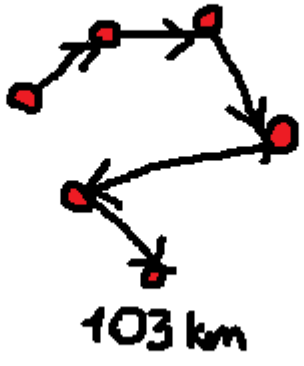


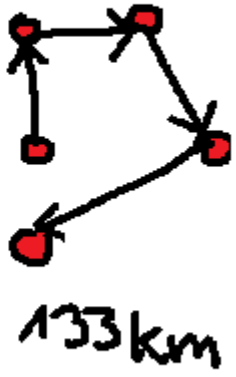
Złożoność obliczeniowa
Notacja dużego O.

Sprzedawca (wolontariusz – leśnik) chce rozwieźć pokarm do pięciu paśników w lesie pokonując jak najmniejszy dystans. Poniżej kilka sposobów odwiedzenia miejscowości (paśników). Leśnik sumuje odległości, a następnie wybiera najkrótszą drogę. Ile istnieje dróg dla 5 miejscowości. Okazuje się, że jest 120 możliwości.

5 miejscowości 120 możliwości
6 miejscowości 720 możliwości
7 miejscowości 5040 możliwości
15 miejscowości 1 307 674 368 000możliwości
30 miejscowości 265 252 859 812 191 058 632 308 480 000 000 możliwości







itd.

Złożoność obliczeniowa
Notacja dużego O.

Skąd te liczby, dlaczego jest tyle możliwości przebycia dróg. Zapis n! mówi ,że jeżeli za n podstawimy liczbę naturalną np. 4 to wyliczenie 4! = 4 * 3 * 2 * 1 = 23. 5! = 5 * 4 * 3 * 2 * 1 = 120. Algorytm sprawdzania wszystkich dróg , a następnie najkrótszej w tym problemie, jest jednym z nierozwiązanych problemów informatyki. Nikt nie zna szybszego algorytmu i uważa się , że opracowanie szybszego jest niemożliwe. Gdyby miejsc odwiedzenia było więcej niż 100, obliczenia trwałyby tak długo , że prędzej doszłoby do śmierci Słońca.

Najszybsze algorytmy

To kilka przykładów złożoności obliczeniowej wyrażonej przy pomocy notacji dużego O. Układając je w kolejności od najszybszego do najwolniejszego:
$O(\log n)$, $O(n)$, $O(n^2)$, $O(n!)$

Zadania z algorytmiki będziemy rozpatrywać pod kątem złożoności obliczeniowej i stosować taki algorytm, który rozwiąże najlepiej dany problem.