Złożoność problemów

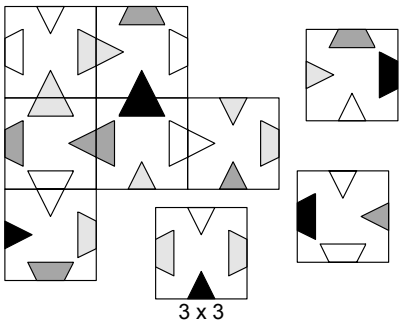
Przykład - wieże Hanoi

Problem jest zamknięty (dolne ograniczenie złożoności = złożoność algorytmu rekurencyjnego lub iteracyjnego) i ma złożoność **O(2^N)**.

Podobno mnisi tybetańscy rozwiązują ten problem dla $N = 64$ i kiedy skończą, to także nasz świat się skończy!

1 ruch na sekundę	⇒	czas wykonania ok. **586 549 402 018 lat**	
1 mln ruchów na sekundę	⇒	czas wykonania ok. **586 549 lat**	☺

Przykład - płaska układanka



Czy istnieje takie ułożenie kwadratu M x M z $N = M^2$ elementów, które zachowuje reguły przystawania boków?

Problem decyzyjny - taki problem algorytmiczny, który polega na znalezieniu odpowiedzi „tak" lub „nie" na postawione pytanie (często jest to odpowiedź na pytanie o istnienie rozwiązania problemu)

Całkowita liczba ułożeń płaskiej układanki wynosi ***N* !**

Dla układanki 5 x 5 oznacza to, że:

1 mln układów na sekundę	⇒	czas sprawdzenia ok. **493 208 500 055 lat**

Wartości niektórych funkcji złożoności:

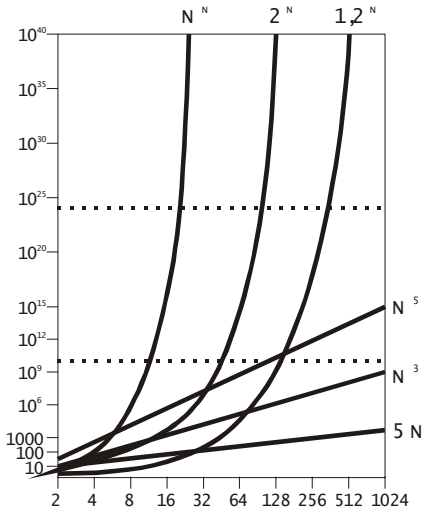
	N				
Funkcja	10	50	100	300	1000
lg N	3	5	6	8	9
N	10	50	100	300	1000
$N * \lg N$	33	282	664	2468	9965
N^2	100	2500	10 000	90 000	1 000 000
N^3	1000	125 000	1 000 000	27 mln (8 cyfr)	1 mld (10 cyfr)
2^N	1024	liczba 16 cyfrowa	liczba 31 cyfrowa	liczba 91 cyfrowa	liczba 302 cyfrowa
N !	3,6 mln (7 cyfr)	liczba 65 cyfrowa	liczba 161 cyfrowa	liczba 623 cyfrowa	∞
N^N	10 mld (11 cyfr)	liczba 85 cyfrowa	liczba 201 cyfrowa	liczba 744 cyfrowa	∞

Dla porównania: liczba protonów w widocznym wszechświecie ma 126 cyfr, liczba mikrosekund od „wielkiego wybuchu" ma 24 cyfry.

Zapotrzebowanie na czas (jeśli jedna instrukcja trwa mikrosekundę)

	N				
Funkcja	10	20	50	100	300
N^2	1/10 000 sekundy	1/2500 sekundy	1/400 sekundy	1/100 sekundy	9/100 sekundy
2^N	1/1000 sekundy	1 sekunda	35,7 lat	400 bilionów stuleci	75 cyfrowa liczba stuleci
N^N	2,8 godziny	3,3 biliony lat	70 cyfrowa liczba stuleci	185 cyfrowa liczba stuleci	728 cyfrowa liczba stuleci

Tempo wzrostu niektórych funkcji złożoności:



Funkcja $f(N)$ jest (asymptotycznie) **ograniczona z góry** przez funkcję $g(N)$,
jeśli $\exists\, N_0\ \forall\, N \geq N_0 : f(N) \leq g(N)$

Jeśli $f(N) \prec g(N)$, to $f(N)$ jest ograniczona z góry przez $g(N)$

Funkcje złożoności dzielimy zgrubnie na:

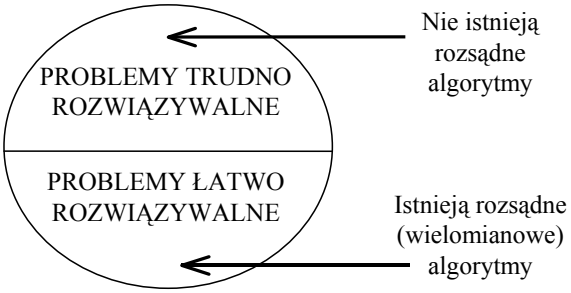
wielomianowe - jeśli istnieje dla nich takie $k < \infty$, że są one ograniczone z góry przez funkcję N^k,

ponadwielomianowe - jeśli takie k nie istnieje (np. wykładnicze).

$$1 \prec \lg\lg N \prec \lg N \prec \sqrt{N} \prec N \prec N \cdot \lg N \prec N^2 \prec N^3 \prec N^{\lg N} \prec 2^N \prec N! \prec N^N \prec 2^{2^N}$$

algorytm wielomianowy = algorytm o złożoności $\mathbf{O}(N^k)$ dla pewnego $k < \infty$

Klasy problemów algorytmicznych (podział zgrubny)



Kilka pytań związanych z czasem rozwiązywania płaskiej układanki:

1. Czy nie można po prostu poczekać na zbudowanie dostatecznie szybkiego komputera?
2. Czy brak rozsądnego algorytmu dla tego problemu nie jest rezultatem braku wiedzy i inwencji informatyków?
3. Czy nie można by wykazać, że dolne ograniczenie złożoności dla tego problemu jest wykładnicze i stwierdzić, że problem jest trudny?
4. Czy nie jest on przypadkiem tak szczególnym, że można go pominąć?

Odpowiedź na pytanie 1.:

	Maksymalna liczba elementów układanki do sprawdzenia w godzinę		
Funkcja	współczesny komputer	komputer 100 razy szybszy	komputer 1000 razy szybszy
N^2	K	$10 * K$	$31{,}6 * K$
2^N	L	$L + 6$	$L + 10$

Odpowiedź na pytanie 4.:

Płaska układanka należy do klasy problemów NPC (**NP-zupełne**), która obejmuje ok. **1000** problemów algorytmicznych o jednakowych cechach:

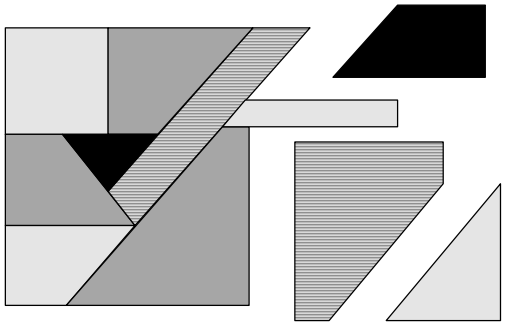
- dla wszystkich istnieją wątpliwe (ponadwielomianowe) rozwiązania
- dla żadnego nie znaleziono rozsądnego (wielomianowego) rozwiązania
- dla żadnego nie udowodniono, że wymaga on czasu wykładniczego
- najlepsze znane dolne ograniczenia złożoności są liniowe, tzn. $\Theta(N)$

Nie wiadomo czy te problemy są trudno, czy łatwo rozwiązywalne!

Na nowe problemy NP-zupełne wciąż napotykamy w kombinatoryce, badaniach operacyjnych, ekonometrii, teorii grafów, logice itd.

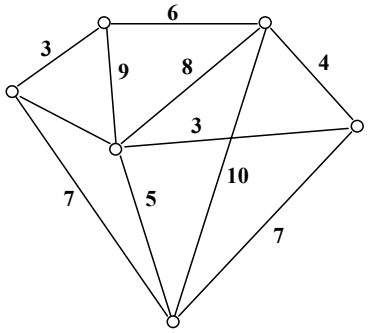
Przykłady problemów NP-zupełnych

Układanki dwuwymiarowe

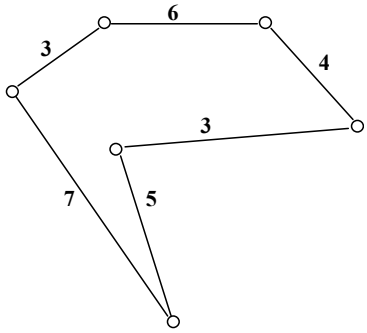


Problem komiwojażera

Problem polega na znalezieniu w sieci połączeń pomiędzy miastami najkrótszej drogi zamkniętej (cyklu), która pozwala odwiedzić każde z miast i powrócić do miasta wyjściowego.
Problem formułowany jest jako poszukiwanie **minimalnego cyklu pełnego w grafie** z wagami krawędzi:



Graf z wagami krawędzi



Minimalny cykl o koszcie 28

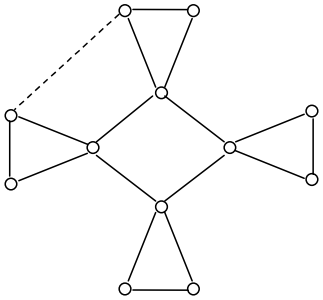
W wersji decyzyjnej problem polega na stwierdzeniu czy istnieje cykl o koszcie nie większym niż podana wartość *K*

Problem pojawia się na przykład przy:

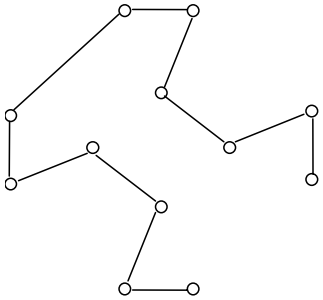
- projektowaniu sieci telefonicznych
- projektowaniu układów scalonych
- planowaniu linii montażowych
- programowaniu robotów przemysłowych

Droga Hamiltona

Problem polega na sprawdzeniu czy w grafie istnieje droga, która przez każdy wierzchołek przechodzi dokładnie raz.

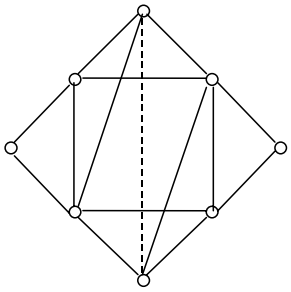


Graf posiadający drogę Hamiltona dopiero po dodaniu krawędzi

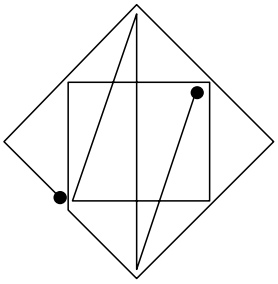


Droga Hamiltona w grafie po dodaniu krawędzi

Dla porównania: sprawdzeniu czy w grafie istnieje droga, która przez każdą krawędź przechodzi dokładnie raz (tzw. problem Eulera) nie jest problemem NP-zupełnym



Graf posiadający drogę Eulera dopiero po dodaniu krawędzi



Droga Eulera w grafie po dodaniu krawędzi

Twierdzenie (Eulera):
W grafie spójnym o parzystej liczbie krawędzi wychodzących z każdego wierzchołka (być może z wyjątkiem dwóch) istnieje droga Eulera.

⇒ problem jest łatwo rozwiązywalny

Przydział ograniczonego miejsca i układanie planu zajęć

Na przykład:
- przydział studentów do pokoi w akademiku z uwzględnieniem różnych ograniczeń
- wypełnianie kontenerów przedmiotami o różnych rozmiarach
- stwierdzanie czy istnieje plan zajęć dopasowujący nauczycieli, klasy i godziny lekcyjne w taki sposób, aby dwie klasy nie miały jednocześnie zajęć z tym samym nauczycielem, nauczyciel nie prowadził w tym samym czasie lekcji w dwóch różnych klasach, dwaj nauczyciele nie prowadzili jednocześnie lekcji w tej samej klasie itd.

Ustalanie czy zdanie logiczne jest spełnialne

Problem polega na stwierdzeniu czy istnieje takie wartościowanie asercji użytych w złożonym zdaniu logicznym, które powoduje, że zdanie to staje się prawdziwe.

Zdanie $\neg(E \Rightarrow F) \wedge (F \vee (D \Rightarrow E))$

staje się prawdziwe dla następującego wartościowania: $E \leftarrow$ PRAWDA, $F \leftarrow$ FAŁSZ, $D \leftarrow$ FAŁSZ
i zatem **jest spełnialne**.

Natomiast zdanie $\neg((D \wedge E) \Rightarrow F) \wedge (F \vee (D \Rightarrow \neg E))$

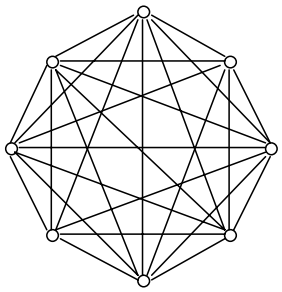
nie jest spełnialne.

Kolorowanie map i grafów 3 kolorami

Kolorowanie mapy 3 kolorami - problem decyzyjny polegający na ustaleniu czy dana mapa może być pokolorowana **3 barwami** tak, aby sąsiednie państwa nie miały tego samego koloru.
- dla **2 barw** problem jest **łatwo rozwiązywalny** - wystarczy sprawdzić czy mapa nie zawiera punktów, w których styka się nieparzysta liczba państw,

- dla **4 barw** problem jest banalny (patrz twierdzenie o czterech barwach)

Kolorowanie grafu - wyznaczenie **minimalnej** liczby barw, którymi można pokolorować wierzchołki danego grafu tak, aby każde dwa wierzchołki bezpośrednio połączone krawędzią miały różne kolory.
Łatwo można skonstruować graf wymagający dowolnie dużej liczby kolorów:



8 barw

Klika - zbiór wierzchołków w grafie połączonych każdy z każdym

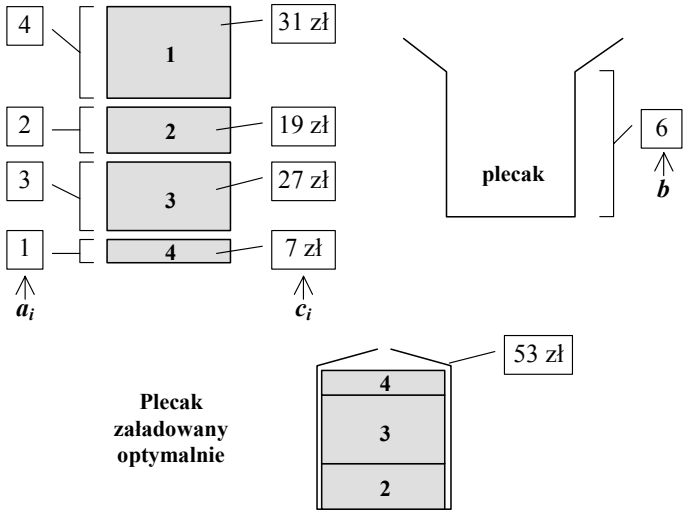
Załadunek plecaka

Problem polega na znalezieniu takiego upakowania przedmiotów do plecaka, które maksymalizuje łączną ich wartość bez przekroczenia pojemności plecaka.
Zapis w wersji optymalizacyjnej:

$$\max \sum_{i=1,..,N} c_i \cdot x_i \text{ , przy ograniczeniach } \sum_{i=1,..,N} a_i \cdot x_i \le b \text{ dla } x_i = 0 \text{ lub } 1$$

Zapis w wersji decyzyjnej: czy dla danego K istnieje takie upakowanie,

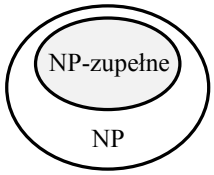
$$\text{że } \sum_{i=1,..,N} c_i \cdot x_i \ge K \text{ i } \sum_{i=1,..,N} a_i \cdot x_i \le b$$



Ogólna charakterystyka problemów z klasy NP

- wymagają sprawdzania rozwiązań częściowych i rozszerzania ich w celu znalezienia rozwiązania ostatecznego; jeśli rozwiązanie częściowe nie da się dalej rozszerzyć, to trzeba powrócić na jakiś wcześniejszy etap i po dokonaniu zmian rozpocząć od nowa
- postępowanie polegające na systematycznym sprawdzeniu wszystkich możliwości wymaga czasu ponadwielomianowego
- jeśli znamy rozwiązanie, to sprawdzenie jego poprawności może być przeprowadzone w czasie wielomianowym!
- dla każdego z problemów istnieje niedeterministyczny („magiczny") algorytm o złożoności wielomianowej; **NP** skrót od ang. *Nondeterministic Polynomial-time*
- są trudno rozwiązywalne, ale stają się łatwo rozwiązywalne, jeśli korzysta się z niedeterministycznej „wyroczni" (por. punkt poprzedni)

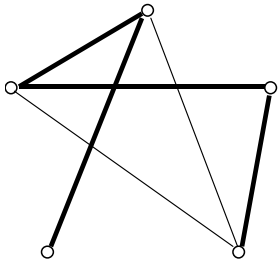
Ogólna charakterystyka problemów NP-zupełnych



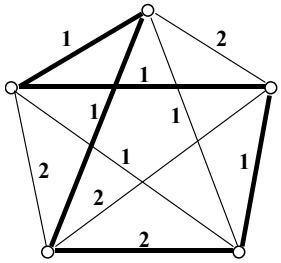
- dla każdego problemu z klasy NP istnieje algorytm o złożoności wielomianowej, którym można go przekształcić do problemu NP-zupełnego (definicja „zupełności problemu"); **NPC** skrót od ang. *Nondeterministic Polynomial-time Complete*
- **każdy problem z tej klasy można przekształcić w czasie wielomianowym do każdego innego!** (wniosek)
- znalezienie algorytmu wielomianowego dla jednego z problemów NPC oznacza możliwość rozwiązania w czasie wielomianowym wszystkich innych! (kolejny wniosek)
- udowodnienie wykładniczego dolnego oszacowania dla jednego z problemów NPC oznacza wykazanie, że żaden z nich nie może być rozwiązany w czasie wielomianowym! (jeszcze jeden wniosek)
- albo wszystkie problemy NPC (NP-zupełne) są łatwo rozwiązywalne albo wszystkie trudno (konkluzja)
- wykazanie, że nowozdefiniowany problem jest NP-zupełny przebiega w dwóch krokach:
  1. trzeba udowodnić, że nowy problem jest klasy NP
  2. trzeba skonstruować algorytm, który w czasie wielomianowym przekształca do jego postaci dowolny znany problem NP-zupełny

Przykłady przekształcania jednego problemu NP-zupełnego w drugi

znalezienie drogi Hamiltona → problem komiwojażera



Droga Hamiltona
dla 5 wierzchołków



Cykl komiwojażera
o długości 6

Istnieje droga Hamiltona w grafie o N wierzchołkach

⇕

Istnieje cykl komiwojażera nie dłuższy niż $N + 1$ w uzupełnionym grafie

kolorowanie mapy 3 kolorami → spełnialność zdania logicznego

Mapa składa się z $P_1, P_2, \ldots, P_N$ państw i mamy trzy kolory C, Z i N.
Asercja $P_K = Z$ oznacza, że państwo P_K jest pokolorowane na zielono.

Zdanie F ma postać $F_1 \wedge F_2$, gdzie F_1 składa się ze zdań

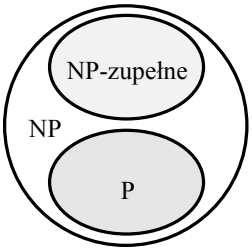
$$(P_K = C \wedge \neg P_K = Z \wedge \neg P_K = N) \vee (\neg P_K = C \wedge P_K = Z \wedge \neg P_K = N) \vee (\neg P_K = C \wedge \neg P_K = Z \wedge P_K = N)$$

powtórzonych dla $K = 1,\ldots,N$ i połączonych koniunkcją, a F_2 składa się ze zdań

$\neg((P_K = C \wedge P_L = C) \vee (P_K = Z \wedge P_L = Z) \vee (P_K = N \wedge P_L = N))$ powtórzonych dla wszystkich par K i L państw sąsiadujących ze sobą i także połączonych koniunkcją.

Klasa NP - problemy posiadające niedeterministyczne algorytmy o złożoności wielomianowej
Klasa P - problemy posiadające zwykłe (deterministyczne) algorytmy o złożoności wielomianowej (łatwo rozwiązywalne)
Klasa NP-zupełne - „wzorcowe" problemy z klasy NP sprowadzalne „szybko" jeden do drugiego

Zawieranie się klas problemów na tym rysunku wynika z sensu notacji O(·), tzn. problemy z klasy P mają złożoność nie gorszą od problemów z klasy NP



Zawieranie się klas problemów na tym rysunku wynika z sensu notacji O(·), tzn. problemy z klasy P mają złożoność nie gorszą od problemów z klasy NP

Algorytmy przybliżone (czyli jak radzić sobie w praktyce z NP)

Np. problem komiwojażera jest NP-zupełny (trudno rozwiązywalny), ale istnieje algorytm o złożoności wielomianowej wyznaczający „niezłe" cykle obchodzące wszystkie wierzchołki grafu.
L_{OPT} - długość minimalnego cyklu (nie potrafimy go szybko wyznaczyć), **Czy P = NP ?**
L_{APR} - długość przybliżonego rozwiązania, które potrafimy wyznaczyć.

Dla miary dobroci rozwiązania przybliżonego: $s_A = \dfrac{L_{APR}}{L_{OPT}}$ istnieje algorytm o złożoności $\mathbf{O}(N^3)$ wyznaczający w najgorszym przypadku cykl minimalny, dla którego $s_A \le 1{,}5$

Przykład algorytmu przybliżonego dla załadunku plecaka

1. Posortuj pakowane przedmioty według nie rosnących wartości $\dfrac{c_i}{a_i}$
2. Pakuj je do plecaka w otrzymanym porządku, dopóki się mieszczą

W przykładowym zadaniu:

$\dfrac{c_1}{a_1} = \dfrac{31}{4} = 7\dfrac{3}{4}$, $\dfrac{c_2}{a_2} = \dfrac{19}{2} = 9\dfrac{1}{2}$, $\dfrac{c_3}{a_3} = \dfrac{27}{3} = 9$, $\dfrac{c_4}{a_4} = \dfrac{7}{1} = 7$ zatem $\dfrac{c_2}{a_2} \ge \dfrac{c_3}{a_3} \ge \dfrac{c_1}{a_1} \ge \dfrac{c_4}{a_4}$

i ta lista wyznacza kolejność pakowania:

$a_2 = 2 \le 6 \quad \rightarrow x_2 = 1$
$a_2 + a_3 = 5 \le 6 \quad \rightarrow x_3 = 1$
$a_2 + a_3 + a_1 = 9 > 6 \quad \rightarrow x_1 = 0$
$a_2 + a_3 + a_4 = 6 \le 6 \quad \rightarrow x_4 = 1$,

złożoność algorytmu = złożoność sortowania = $= \mathbf{O}(N \cdot \lg N)$

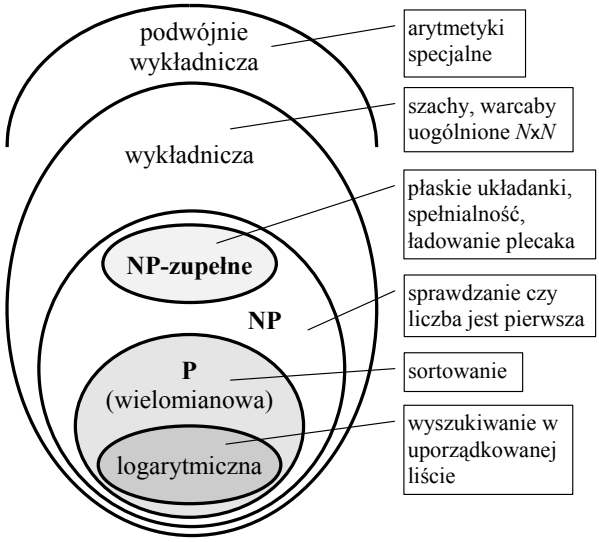
$c_2 + c_3 + c_4 = 53 \Rightarrow s_A = 1$

W najgorszym przypadku algorytm daje upakowanie o $s_A \le 2$

Uwagi uzupełniające:

- po raz pierwszy wykazano NP-zupełność dla problemu spełnialności zdania logicznego (Cook w 1971 r.)
- są problemy, dla których udowodniono, że należą do klasy NP, ale nie są ani NP-zupełne, ani nie należą do klasy P , np. sprawdzenie czy dana liczba jest liczbą pierwszą
- są problemy, których złożoność ponadwielomianową można udowodnić przez podanie dolnych ograniczeń czasowych (i to nie tylko takie, jak wieże Hanoi, dla których z góry wiadomo ile iteracji wykona algorytm), np. stwierdzenie czy dla danej konfiguracji w uogólnionych szachach N x N istnieje strategia wygrywająca dla jednego z przeciwników
- są problemy, dla których pokazano podwójnie wykładnicze $\Theta(2^{2^N})$ dolne ograniczenia czasowe
- są problemy algorytmiczne, dla których udowodniono, że mają ponadwielomianowe dolne ograniczenia złożoności pamięciowej
- są ciekawe przypadki problemów, dla których efektywne w praktyce algorytmy mają złożoność ponadwielomianową, choć znaleziono dla nich algorytm wielomianowy sprawujący się jednak w większości zadań praktycznych wyraźnie gorzej, np. zadanie programowania liniowego z algorytmem sympleksowym (wykładniczym) i algorytmem elipsoidalnym (wielomianowym)

Klasy złożoności algorytmów



Uwaga:
Zawieranie się klas problemów na tym rysunku zachodzi w sensie notacji O(·)