Politechnika Wrocławska

Struktury danych i złozoność obliczeniowa

Prof. dr hab. inż. Jan Magott

Formy zajęć:

- Wykład 1 godz.,
- Ćwiczenia 1 godz.,
- Projekt 2 godz. .

Adres strony z materiałami do wykładu:

http://www.zio.iiar.pwr.wroc.pl/sdizo.html

Struktury danych wchodzące w zakres kursu:

- Tablice,
- Listy,
- Kolejki,
- Stosy,
- Grafy,
- Drzewa binarne,
- Kopce,
- Tablice haszujące.

Złożoność obliczeniowa

Wg teorii złożoności obliczeniowej **efektywnym** jest algorytm o złożoności czasowej ograniczonej od góry przez wielomian od rozmiarów problemu.

Algorytmy o wykładniczej złożoności obliczeniowej uważane są za **nieefektywne**.

Problem dodawania macierzy kwadratowych

Dane:

Macierze $A_{n\times n}$, $B_{n\times n}$

Zadanie:

Wyznaczyć macierz

$$C_{n\times n} = A_{n\times n} + B_{n\times n}$$

Algorytm dodawania macierzy kwadratowych

Algorytm wyrażony w języku zawierającym elementy języka naturalnego i formalnego

Dla każdej pary $< i, j > \in \{\overline{1,n}\} \times \{\overline{1,n}\}$ wykonaj

$$c_{ij} = a_{ij} + b_{ij}$$

Oznaczenia:

$\{\overline{1,n}\} = \{1,2,\dots,n\}$,

$X \times Y$ - iloczyn kartezjański zbiorów X, Y

Złożoność obliczeniowa algorytmu

$$\approx k \cdot n^2$$

n jest rozmiarem problemu.

Rozmiar problemu dla danych w postaci zbioru

$$\{x_1, x_2, \ldots, x_n\}$$

Przykłady rozmiaru:

- Liczba elementów czyli n,
- Liczba znaków w dziesiętnym kodowaniu n elementów,
- Liczba znaków w dwójkowym kodowaniu n elementów,
- Liczba znaków w jedynkowym kodowaniu n elementów.

Kodowanie

$$(7)_{10} = (111)_2 = (1111111)_1$$

Przykłady problemów o więcej niż jednym rozmiarze danych:

Problem mnożenia macierzy nie kwadratowych – 3 rozmiary,

Problem najkrótszej drogi w grafie – rozmiarami mogą być: liczba wierzchołków, liczba łuków.

Problem mnożenia macierzy kwadratowych

Dane:

Macierze $A_{n\times n}$, $B_{n\times n}$

Zadanie:

Wyznaczyć macierz

$$C_{n\times n} = A_{n\times n} \cdot B_{n\times n}$$

Algorytm mnożenia macierzy kwadratowych

Algorytm wyrażony w języku zawierającym elementy języka naturalnego i formalnego

Dla każdej pary $< i, j > \in \{\overline{1,n}\} \times \{\overline{1,n}\}$ wykonaj

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Złożoność obliczeniowa algorytmu

$$\approx k \cdot n^3$$

Złożoność algorytmów

Algorytmu dodawania macierzy kwadratowych

$$\approx k \cdot n^2$$

Algorytmu mnożenia macierzy kwadratowych

$$\approx k \cdot n^3$$

Powyższe funkcje są wielomianami, a zatem te algorytmy są uważane za efektywne.

Problem wyznaczania elementu maksymalnego w zbiorze

$$max = max\{A1, A2, \dots, An\}$$

Pseudokod algorytmu

```
max(A1, A2,…, to An integers);
max := A1;
for i:= 2 to n do
        if max < Ai then max := Ai;
return max {max is the largest element}
```

Złożoność obliczeniowa algorytmu

$$\approx k \cdot n$$

Problem sortowania zbioru

$$\{x_1, x_2, \ldots, x_n\}$$

niemalejąco.

Idea algorytmu sortowania bąbelkowego

Algorytm opiera się na zasadzie: każda liczba jest mniejsza lub równa od liczby maksymalnej. Porównując kolejno liczby można wyznaczyć największą z nich. Następnie ciąg częściowo posortowany (mający na końcu posortowane liczby maksymalne), można skrócić o te liczby i ponowić szukanie maksimum, już bez elementów posortowanych i tak długo, aż zostanie nam jeden element. Otrzymane kolejne maksima są coraz mniejsze przez co ciąg jest uporządkowany.

Algorytm sortowania bąbelkowego w kodzie C

```
void bubblesort(int table[], int size)
{
        int i, j, temp;
        for (i = 0; i<size; i++)
    {
                for (j=0; j<size-1-i; j++)
                {
                        if (table[j] > table[j+1])
                        {
                                temp = table[j+1];
                                table[j+1] = table[j];
                                table[j] = temp;
                        }
                }
    }
}
```

Złożoność algorytmu sortowania bąbelkowego

$$\approx k \cdot n^2$$

Problem podziału zbioru

Dane:

- $X = \{x_1, \dots, x_i, \dots, x_k\}$ - zbiór k elementów $x_i \in N_+$, gdzie $N_+ = \{1,2, \dots\}$,
- $B \in N_+$,
- $\sum_{i=1}^{k} x_i = 2B$.

Pytanie:

Czy istnieje podzbiór $X_1 \subset X$ taki, że

$$\sum_{x_i \in X_1} x_i = B \ ?$$

Problem:

Czy istnieje algorytm wielomianowy dla problemu podziału zbioru?

Problem:

Czy istnieje algorytm wielomianowy dla problemu podziału zbioru?

Udowodniono:

Problem podziału zbioru jest tzw. Problemem NP-zupełnym czyli wydaje się prawie pewne, że algorytmu wielomianowego nie uda się zbudować.

Intuicyjne pojmowanie klas P i NP

Klasa P zawiera te wszystkie problemy decyzyjne, dla których znaleziono wielomianowe algorytmy ich rozwiązania.

Klasa NP zawiera te wszystkie problemy decyzyjne, dla których znaleziono ponadwielomianowe algorytmy ich rozwiązania (w wielomianowym czasie można odgadnąć rozwiązanie i sprawdzić czy to rozwiązanie daje odpowiedź "tak").

Problem decyzyjny π_1 jest nazywany ***NP-zupełnym***, jeśli:

1. $\pi_1 \epsilon NP$,
2. Dla każdego innego problemu decyzyjnego $\pi_2 \epsilon NP$ jest $\pi_2 \propto \pi_1$.

Zatem, jeśli istniałby algorytm wielomianowy do rozwiązywania jakiegokolwiek problemu NP-zupełnego, to każdy problem z klasy NP (w tym również problemy NP-zupełne) mógłby być rozwiązany za pomocą algorytmu wielomianowego.

Z bezskuteczności poszukiwań algorytmu wielomianowego dla któregokolwiek problemu NP-zupełnego wynika, że **prawie na pewno wszystkie problemy NP-zupełne można rozwiązać tylko przy użyciu algorytmów ponadwielomianowych**.

Decyzyjny problem szeregowania zadań niezależnych na dwu maszynach (procesorach)

Dane:

- $T = \{t_1, \ldots, t_i, \ldots, t_k\}$ – zbiór czasów wykonania k zadań niezależnych (bez ograniczeń kolejności wykonywania) gdzie $t_i \in N_+$,
- $B \in N_+$,
- $\sum_{i=1}^{k} t_i = 2B$.

Pytanie:

Czy można te zadania tak rozłożyć na dwie maszyny, że sumaryczny czas wykonania będzie równy B ?

Czy rozwiązanie jednego z dwu problemów:

Decyzyjnego problemu szeregowania zadań niezależnych na dwu maszynach

i

Problemu podziału zbioru

można wyznaczyć na postawie rozwiązania drugiego?

Powyższe pytanie dotyczy redukcji (transformacji) jednego problemu do drugiego.

Problem podziału zbioru

Dane:

- $X = \{x_1, \dots, x_i, \dots, x_k\}$ - zbiór k elementów $x_i \in N_+$, gdzie $N_+ = \{1,2, \dots\}$,
- $B \in N_+, \quad \sum_{i=1}^{k} x_i = 2B.$

Pytanie:

Czy istnieje podzbiór $X_1 \subset X$ taki, że $\sum_{x_i \in X_1} x_i = B$?

Decyzyjny problem szeregowania zadań niezależnych na dwu maszynach (procesorach)

Dane:

- $T = \{t_1, \dots, t_i, \dots, t_k\}$ - zbiór czasów wykonania k zadań niezależnych (bez ograniczeń kolejności wykonywania) gdzie $t_i \in N_+$,
- $B \in N_+, \ \sum_{i=1}^{k} t_i = 2B.$

Pytanie:

Czy można te zadania tak rozłożyć na dwie maszyny, że sumaryczny czas wykonania będzie równy B ?

Czy rozwiązanie jednego z dwu problemów:

Decyzyjnego problemu szeregowania zadań niezależnych na dwu maszynach

i

Problemu podziału zbioru

można wyznaczyć na postawie rozwiązania drugiego?

Powyższe pytanie dotyczy redukcji (transformacji) jednego problemu do drugiego.

Jakie ograniczenie narzucilibyśmy na złożoność obliczeniową transformacji?

Problem charakteryzowany jest poprzez *Dane* (dane wejściowe)

i *Zadanie* do wykonania lub *Pytanie*.

Problem dodawania macierzy kwadratowych

Dane:

Macierze $A_{n\times n}$, $B_{n\times n}$

Zadanie:

Wyznaczyć macierz

$$C_{n\times n} = A_{n\times n} + B_{n\times n}$$

Problem podziału zbioru

Dane:

- $X = \{x_1, \dots, x_i, \dots, x_k\}$ - zbiór k elementów $x_i \in N_+$, gdzie $N_+ = \{1,2, \dots\}$,
- $B \in N_+$, $\sum_{i=1}^{k} x_i = 2B$.

Pytanie:

Czy istnieje podzbiór $X_1 \subset X$ taki, że $\sum_{x_i \in X_1} x_i = B$?

Problemy rozstrzygalne i nierozstrzygalne

Problem rozstrzygalny (rozwiązywalny) to problem, dla którego istnieje algorytm znajdujący rozwiązanie w skończonej liczbie kroków.

Problem nierozstrzygalny (nierozwiązywalny) to problem, dla którego udowodniono, że nie istnieje algorytm znajdujący rozwiązanie w skończonej liczbie kroków.

Złożoność obliczeniową badamy dla problemów rozstrzygalnych.

Algorytm to procedura do rozwiązywania problemu.

Algorytm może być wyrażony w:

- Języku naturalnym,
- Języku formalnym,
- Języku zawierającym konstrukcje języka naturalnego i formalnego,
- Pseudokodzie,
- Języku programowania.

Złożoność obliczeniowa algorytmu a złożoność problemu

Algorytm rozwiązujący problem może mieć złożoność obliczeniową logarytmiczną, liniową, wielomianową, wykładniczą i wiele innych.

Analizowana jest złożoność czasowa i pamięciowa.

Jeśli problem jest NP-zupełny, to prawie na pewno nie zostanie znaleziony wielomianowy algorytm rozwiązujący.

Nie można mówić, że dla problemu NP-zupełnego najprawdopodobniej nie zostanie znaleziony wielomianowy algorytm rozwiązujący.

Następny slajd pochodzi z
Krajowych Ram Kwalifikacyjnych kursu

Struktury danych i złożoność obliczeniowa

Forma zajęć - wykład

Wy1	Zajęcia organizacyjne: program, wymagania, literatura.	1
Wy2	Wprowadzenie do teorii złożoności obliczeniowej – kodowanie danych wejściowych.	2
Wy3, Wy4	Eksplozja kombinatoryczna. Algorytmy wielomianowe i ponadwielomianowe. Klasy złożoności problemów decyzyjnych (P, NP, NP-zupełne i silnie NP-zupełne). Relacja pomiędzy NP-zupełnością i NP-trudnością.	4
Wy5, Wy6	Przykłady problemów wielomianowo rozwiązywalnych i NP-zupełnych. Transformacja wielomianowa. Zarys przeprowadzania dowodów NP-zupełności.	4
Wy7	Algorytmy pseudowielomianowe. Problemy liczbowe i silna NP-zupełność.	2
Wy8	Kolokwium	2
	Suma godzin	15

Oceny (F – formująca (w trakcie semestru), P – podsumowująca (na koniec semestru)		Sposób oceny osiągnięcia efektu kształcenia
F1		Odpowiedzi ustne, Wyniki kolokwiów cząstkowych.
F2		Wyniki realizacji zadań projektowych
F3		Kolokwium pisemne
$P = 0,25 \cdot F1 + 0,25 \cdot F2 + 0,5 \cdot F3$		

Literatura

[1] T. Cormen, C.E. Leiserson, R.L. Rivest, „Wprowadzenie do algorytmów”, WNT 2003.

[2] J. Błażewicz, „Problemy optymalizacji kombinatorycznej”, PWN, Warszawa 1996.

[3] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Co., New York, 1979.

Instancją I problemu π jest konkretny problem ze zdefiniowanymi wszystkimi danymi wejściowymi.

Zbiór instancji problemu π oznaczać będziemy symbolem D_π .

Należy rozróżniać problemy i ich instancje.

Problem podziału zbioru

Dane:

- $X = \{x_1, \dots, x_i, \dots, x_k\}$ - zbiór k elementów $x_i \in N_+$, gdzie $N_+ = \{1,2, \dots\}$,
- $B \in N_+, \quad \sum_{i=1}^{k} x_i = 2B.$

Pytanie:

Czy istnieje podzbiór $X_1 \subset X$ taki, że $\sum_{x_i \in X_1} x_i = B$?

Instancja problemu podziału zbioru

Dane:

- $X = \{x_1, \dots, x_i, \dots, x_k\} = \{3,11,7,23,18\}$ - zbiór pięciu elementów,
- $B = 31, \quad \sum_{i=1}^{5} x_i = 2 \cdot B.$

Pytanie:

Czy istnieje podzbiór $X_1 \subset X$ taki, że $\sum_{x_i \in X_1} x_i = B$?

Kodowanie danych wejściowych

Sposób kodowania danych wejściowych ma wpływ na rozmiar problemu (długość danych wejściowych)

Czy każdy z następujących sposobów kodowania liczb:

- Dziesiętny,
- Binarny,
- Jedynkowy

jest „właściwy” ?

Alfabet jest skończonym zbiorem symboli.

Alfabet oznaczamy symbolem $\sum$.

Przykłady

Alfabet dziesiętny $\sum = \{0,1,2,\ldots,8,9,\sqcup\}$, gdzie $\sqcup$ jest separatorem.

Alfabet dwójkowy $\sum = \{0,1,\sqcup\}$.

Alfabet jedynkowy $\sum = \{1,\sqcup\}$.

Słowem alfabetu $\sum$ nazywamy skończony ciąg symboli alfabetu $\sum \backslash \{\sqcup\}$.

Szczególnym przypadkiem jest słowo puste czyli nie zawierające żadnego symbolu

Przykłady

Słowa dziesiętne: 536, 1429, 374502

Słowa dwójkowe: 10, 0, 1001011

Słowa jedynkowe: 111, 11111, 1, 11111111

Słownikiem alfabetu $\sum$ jest zbiór wszystkich słów alfabetu $\sum \backslash \{\sqcup\}$.

Przykłady

Słownik dziesiętny:

{0,1,2,…,9,10,11,…,99,100, 101, …}

Słownik dwójkowy:

{0,1,10,11,100,101, …}

Słownik jedynkowy:

{1,11,111,1111,…}

Językiem L alfabetu $\sum$ jest skończony ciąg słów tego alfabetu oddzielonych separatorami $\sqcup$.

Przykłady

Język dziesiętny: 402$\sqcup$31$\sqcup$58

Język dwójkowy: 1011 $\sqcup$101111 $\sqcup$11 $\sqcup$1000

Język jedynkowy: 11$\sqcup$1111$\sqcup$1$\sqcup$111$\sqcup$111