Wprowadzenie do Algorytmiki. Programowanie Dynamiczne.

Artur Laskowski

16 grudnia 2021, Poznań

Zadanie QAP

15 minut

Musi być zaliczony

Zadanie QAP

- Dane jest n fabryk i n odbiorców
- Dana jest macierz odległości pomiędzy fabrykami i odbiorcami
- Każdą fabrykę należy przyporządkować do innego odbiorcy, tak aby suma odległości była minimalna

Zadanie QAP

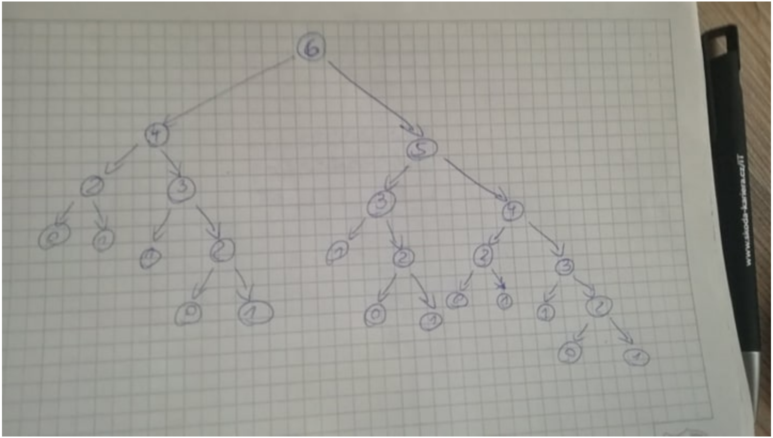
	1	2	3	4	5
A	5	3	7	4	9
B	8	6	3	5	4
C	7	6	2	4	7
D	5	6	5	7	3
E	4	3	8	5	6

Liczby Fibonacciego

- Każda liczba jest sumą dwóch poprzednich
- $F_0 = 1, \qquad F_1 = 1, \qquad F_n = F_{n-1} + F_{n-2}$
- Liczby te często występują w naturze

Liczby Fibonacciego

Wywołania funkcji dla obliczania n-tej liczby Fibinacciego.



Rozwiązanie naiwne

- Przez zastosowanie rekurencji wiele podproblemów liczymy wielokrotnie
- Otrzymujemy wykładniczą złożoność obliczeniową
- Możemy zapisać sobie rozwiązanie raz obliczonego podproblemu, aby nie musieć robić tego ponownie!

Rozwiązanie przy pomocy programowania dynamicznego

```
int dp[100];

int fibonacci(int id){
    dp[0] = dp[1] = 1;

    if(dp[id] == 0) {
        dp[id] = fibonacci(id-1) + fibonacci(id-2);
    }

    return dp[id];
}
```

Rozwiązanie iteracyjne

```
int dp[100];

int fibonacci(int id) {
    dp[0] = dp[1] = 1;
    for(int i = 2; i <= id; ++i) {
        dp[i] = dp[i-1] + dp[i-2];
    }
    return dp[i];
}
```

- Oba rozwiązania działają w czasie liniowym
- Rozwiązanie iteracyjne ma znacząco mniejszą stałą
- Rozwiązanie rekurencyjne bywa łatwiejsze w zrozumieniu oraz implementacji
- Często warto spróbować rekurencyjnie, a jeśli nie działa to dopiero iteracyjnie

Programowanie Dynamiczne - schemat

- Wyznaczanie struktury optymalnego rozwiązania $(F[n])$
- Rekurencyjne zdefiniowanie funkcji optymalnego rozwiązania $F[n] = F[n-1] + F[n-2]$
- Obliczenie wartości funkcji metodą wstępującą $(F[1], F[2], ...)$
- Konstruowanie rozwiązania przy pomocy wcześniejszych obliczeń

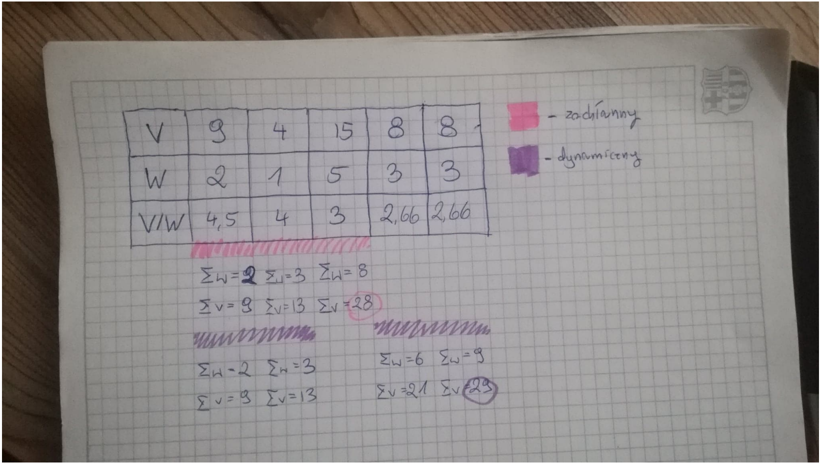
Problem plecakowy

- Dane jest n przedmiotów
- Każdy przedmiot ma wartość v_i oraz wagę w_i
- Dany jest plecak, który może pomieścić przedmioty o maksymalnej wadze m
- Jaka jest maksymalna wartość przedmiotów, które można spakować?

Algorytm zachłanny

- Zabierzemy te przedmioty, które mają najlepszą proporcję wartości do wagi
- Nie zawsze działa

Algorytm zachłanny



Rozwiązanie dynamiczne

- Rozwiązujemy podproblemy, aby dojść do oryginalnego zadania
- Podproblem (i, j) oznacza, że używamy przedmiotów 1..i oraz plecak ma pojemność j
- T(i, j) oznacza maksymalną sumę wartości przedmiotów, które udało się zmieścić w takim plecaku

Rozwiązanie dynamiczne

- Wybieramy przedmiot i-ty, wtedy:

$$T(i,j) = T(i-1, j-w_i) + v_i$$

- Nie wybieramy przedmiotu i-tego, wtedy:

$$T(i,j) = T(i-1, j)$$

Rozwiązanie dynamiczne

Przykład na tablicy!

Implementacja

```
for(int i = 0; i <= n; ++i) {
    for(int j = 0; j <= m; ++j) {
        t[i][j] = t[i-1][j];
        if(w[i] <= j) {
            t[i][j] = max(t[i][j], t[i-1][j-w[i]] + v[i]);
        }
    }
}
```

Rozwiązanie dynamiczne

- Złożoność obliczeniowa – $O(n \cdot m)$
- Złożoność pseudowielomianowa – dla m o jeden bit większego algorytm działa dwa razy wolniej
- Jeżeli nie chcemy wiedzieć, które przedmioty są w plecaku wystarczy pamiętać tylko ostatni wiersz – złożoność pamięciowa $O(m)$ zamiast $O(n \cdot m)$

Alternatywne problemy

- Problem ciągły – z każdego przedmiotu możemy wziąć jego dowolnie małą część:
- Algorytm zachłanny działa - $O(nlogn)$

Alternatywne problemy

- Każdy przedmiot występuje w nieskończonej liczbie sztuk:
- Dodatkowe $Max(..., T(i, j - w[i]) + v[i]))$

Inne problemy

- Najdłuższy, wspólny, niekoniecznie spójny podciąg dwóch ciągów

$$B, D, C, A, B, A$$

$$A, B, C, B, D, A, B$$

Inne problemy

- Wydawanie reszty

$$Dostpne : 1, 1, 1, 5, 10, 20$$

$$Wydajemy : 17, 19$$

Inne zadania

Laboratoria

https://www.hackerrank.com/wda-07-2021