Wstęp do programowania

Algorytmy zachłanne, algoritme Dijkstry

Paweł Daniluk

Wydział Fizyki

Jesień 2013

Algorytmy zachłanne

Strategia polegająca na budowaniu rozwiązania problemu przez dokonywanie lokalnie optymalnych wyborów.

Algorytmy zachłanne

Strategia polegająca na budowaniu rozwiązania problemu przez dokonywanie lokalnie optymalnych wyborów.

W większości przypadków ta strategia nie prowadzi do optymalnego rozwiązania

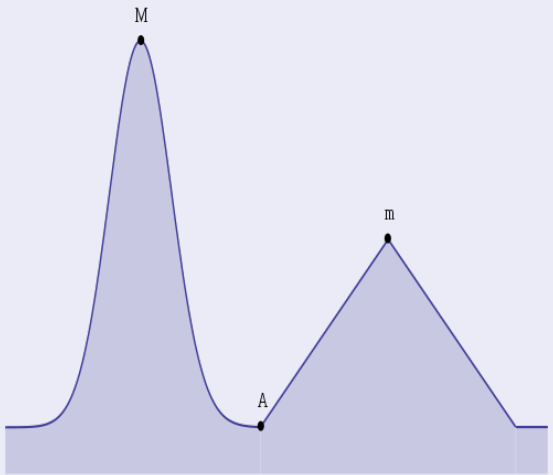
Algorytmy zachłanne

Strategia polegająca na budowaniu rozwiązania problemu przez dokonywanie lokalnie optymalnych wyborów.

W większości przypadków ta strategia nie prowadzi do optymalnego rozwiązania

Analogia

Podchodzenie pod górę:



Algorytmy zachłanne c.d.

Problem wydawania reszty

W jaki sposób dysponując monetami o danych nominałach (np. 5/2/1) wydać resztę używając do tego minimalnej liczby monet.

Rozwiązanie zachłanne polega na wydawaniu monety o największym możliwym nominale. Np.:

$$13 = 5 + 5 + 2 + 1$$

Algorytmy zachłanne c.d.

Problem wydawania reszty

W jaki sposób dysponując monetami o danych nominałach (np. 5/2/1) wydać resztę używając do tego minimalnej liczby monet.

Rozwiązanie zachłanne polega na wydawaniu monety o największym możliwym nominale. Np.:

$$13 = 5 + 5 + 2 + 1$$

Metoda zachłanna nie działa dla każdego zestawu monet. Np. dla monet 5/2:

$$6 = 5 + ?$$

Zastosowania

Własność optymalnej podstruktury

Optymalne rozwiązanie problemu jest funkcją optymalnych rozwiązań podproblemów.

Rozwiązania dokładne (optymalne)

- ortogonalizacja układu wektorów
- kodowanie Huffmana
- problem wyboru czynności
- algorytm Dijkstry

Zastosowania

Własność optymalnej podstruktury

Optymalne rozwiązanie problemu jest funkcją optymalnych rozwiązań podproblemów.

Rozwiązania dokładne (optymalne)

- ortogonalizacja układu wektorów
- kodowanie Huffmana
- problem wyboru czynności
- algorytm Dijkstry

Heurystyki

Algorytmy zachłanne są wydajne. Często się ich używa do znajdowania rozwiązań przybliżonych.

Algorytm Dijkstry

Problem

Dany jest graf nieskierowany z nieujemnymi wagami krawędzi. Wyznaczyć ścieżkę pomiędzy danymi wierzchołkami o najmniejszej wadze.

Przykładowe zastosowanie

Dane są połączenia drogowe pomiędzy miastami. Wyznaczyć najkrótszą trasę.

Algorytm Dijkstry

Algorytm

Przez s oznaczamy wierzchołek źródłowy, $w(i,j)$ to waga krawędzi (i,j).

1. Stwórz tablicę d odległości od źródła dla wszystkich wierzchołków grafu. Na początku $d[s] = 0$, zaś $d[v] = \infty$ dla wszystkich pozostałych wierzchołków.
2. Utwórz kolejkę priorytetową Q wszystkich wierzchołków grafu. Priorytetem kolejki jest aktualnie wyliczona odległość od wierzchołka źródłowego s.
3. Dopóki kolejka nie jest pusta:
   1. Usuń z kolejki wierzchołek u o najniższym priorytecie (wierzchołek najbliższy źródła, który nie został jeszcze rozważony)
   2. Dla każdego sąsiada v wierzchołka u dokonaj relaksacji poprzez u: jeśli $d[u] + w(u,v) < d[v]$ (poprzez u da się dojść do v szybciej niż dotychczasową ścieżką), to $d[v] := d[u] + w(u,v)$.

Na końcu tablica d zawiera najkrótsze odległości do wszystkich wierzchołków.

Zadanie 1 – Segment o maksymalnej sumie

Zadanie

Napisz program znajdujący w zadanej tablicy A typu int[N], maksymalną sumę segmentu (spójnego fragmentu tablicy). Przyjmujemy, że segment pusty ma sumę 0.

Zadanie 1 – Segment o maksymalnej sumie

Zadanie

Napisz program znajdujący w zadanej tablicy A typu int[N], maksymalną sumę segmentu (spójnego fragmentu tablicy). Przyjmujemy, że segment pusty ma sumę 0.

Wskazówki

- Najprostsze rozwiązanie to dla wszystkich możliwych segmentów policzyć ich sumę.

Zadanie 1 – Segment o maksymalnej sumie

Zadanie

Napisz program znajdujący w zadanej tablicy A typu int[N], maksymalną sumę segmentu (spójnego fragmentu tablicy). Przyjmujemy, że segment pusty ma sumę 0.

Wskazówki

- Najprostsze rozwiązanie to dla wszystkich możliwych segmentów policzyć ich sumę.
- W powyższym rozwiązaniu sumę pewnych segmentów liczy się wiele razy. Lepiej dla danego początku l obliczać po kolei sumy coraz dłuższych segmentów zaczynających się w l.

Zadanie 1 – Segment o maksymalnej sumie

Zadanie

Napisz program znajdujący w zadanej tablicy A typu int[N], maksymalną sumę segmentu (spójnego fragmentu tablicy). Przyjmujemy, że segment pusty ma sumę 0.

Wskazówki

- Najprostsze rozwiązanie to dla wszystkich możliwych segmentów policzyć ich sumę.
- W powyższym rozwiązaniu sumę pewnych segmentów liczy się wiele razy. Lepiej dla danego początku l obliczać po kolei sumy coraz dłuższych segmentów zaczynających się w l.
- Niech Pref(i) oznacza sumę elementów tablicy od 1 do i włącznie. Suma segmentu od l do p to oczywiście Pref(p) - Pref(l-1). Maksymalną sumę segmentu kończącego się w p uzyskamy odejmując od Pref(p) minimalne Pref(i) gdzie i przebiega [1..p].

Zadanie 1 – Segment o maksymalnej sumie c.d.

Przydatna sztuczka w tym problemie – sumy prefiksowe

$$P(k) = \sum_{i=0}^{k-1} T_i$$

$$\sum_{i=s}^{e-1} T_i = P(e) - P(s)$$

$$\max_{0 \le s \le e \le N} \sum_{i=s}^{e-1} T_i = \max_{0 \le s \le e \le N} P(e) - P(s) = \max_{0 \le e \le N} P(e) - \min_{0 \le s \le e} P(s)$$

Zadanie 2 – Sortowanie szybkie (*quick sort*)

Algorytm

Algorytm działa rekursywnie - wybierany jest pewien element tablicy, tzw. element osiowy, po czym na początek tablicy przenoszone są wszystkie elementy mniejsze od niego, na koniec wszystkie większe, a w powstałe między tymi obszarami puste miejsce trafia wybrany element. Potem sortuje się osobno początkową i końcową część tablicy. Rekursja kończy się, gdy kolejny fragment uzyskany z podziału zawiera pojedynczy element, jako że jednoelementowa podtablica nie wymaga sortowania.
Niech l oznacza indeks pierwszego, r – ostatniego elementu sortowanego fragmentu tablicy, zaś i – indeks elementu, na którym tablica została podzielona.

Zadanie 2 – Sortowanie szybkie (*quick sort*) c.d.

Pseudokod

```
PROCEDURE Quicksort(l, r)
  BEGIN
    IF l < r THEN { jeśli fragment dłuższy niż 1 element }
      BEGIN
        i = PodzielTablice(l, r); { podziel i zapamiętaj punkt
        Quicksort(l, i-1);        { posortuj lewą część }
        Quicksort(i, r);          { posortuj prawą część }
      END
  END
```

Zadanie 3 – Smith-Waterman

Napisz program obliczający optymalne globalne uliniowienie podanych sekwencji.

Strona wykładu

http://bioexploratorium.pl/wiki/Wstęp_do_programowania_
_2013z