Lista zadań. Nr 1. 18 lutego 2012

Algorytmy i Struktury Danych

IIUWr. II rok informatyki.

1. (0 pkt) Przeczytaj notatkę numer 1 umieszczoną na stronie wykładu.
2. (0 pkt) Przypomnij sobie algorytm sortowania bąbelkowego. Zapisz go w notacji zbliżonej do tej, której używaliśmy na wykładzie. Porównaj go z algorytmami *InsertSort* i *SelectSort* stosując podane na wykładzie kryteria.
3. (1pkt) Rozwiąż zadanie z Listy Powitanej na Themis
4. (1pkt) Udowodnij, że algorytm mnożenia liczb *"po rosyjsku"* jest poprawny. Jaka jest jego złożoność czasowa i pamięciowa przy:
   - jednorodnym kryterium kosztów,
   - logarytmicznym kryterium kosztów?
5. (2pkt) Pokaż, w jaki sposób algorytm "macierzowy" obliczania n-tej liczby Fibonacciego można uogólnić na inne ciągi, w których kolejne elementy definiowane są liniową kombinacją skończonej liczby elementów wcześniejszych. Następnie uogólnij swoje rozwiązanie na przypadek, w którym n-ty element ciągu definiowany jest jako suma kombinacji liniowej skończonej liczby elementów wcześniejszych oraz wielomianu zmiennej n.
6. (1pkt) Niech u i v będą dwoma wierzchołkami w grafie nieskierowanym $G = (V, E; c)$, gdzie $c : E \to R_+$ jest funkcją wagową. Mówimy, że droga z $u = u_1, u_2, \ldots, u_{k-1}, u_k = v$ z u do v jest sensowna, jeśli dla każdego $i = 2, \ldots, k$ istnieje droga z u_i do v krótsza od każdej drogi z u_{i-1} do v (przez długość drogi rozumiemy sumę wag jej krawędzi).

   Ułóż algorytm, który dla danego G oraz wierzchołków u i v wyznaczy liczbę sensownych dróg z u do v.
7. (2pkt) Wykonanie operacji $insert(L, k)$ wstawiającej liczbę k do uporządkowanej rosnącą listy L, może wymagać przejrzenia $\Omega(n)$ elementów listy. Pokaż, w jaki sposób koszt tej operacji można zredukować do $O(\sqrt{n})$, wykorzystując $O(\sqrt{n})$ dodatkowych komórek pamięci.
8. (2pkt) Ułóż algorytm, który dla zadanego acyklicznego grafu skierowanego G znajduje długość najdłuższej drogi w G. Następnie zmodyfikuj swój algorytm tak, by wypisywał drogę o największej długości (jeśli jest kilka takich dróg, to Twój algorytm powinien wypisać dowolną z nich).

Zadania dodatkowe - nie będą rozwiązywane w czasie ćwiczeń

1. (1pkt) Co stałoby się z mocą obliczeniową maszyny RAM gdyby instrukcje ADD i MULT zostały usunięte z repertuaru instrukcji? Jak zmieniłby się koszt obliczeń?
2. (1pkt) Pokaż, że dla każdego programu maszyny RAM istnieje równoważny program maszyny RAM (tj. taki, który dla tych samych danych produkuje te same wyniki) używający nie więcej niż 2^{14} komórek pamięci.
3. (0pkt) Przypomnij sobie notację asymptotyczną dla rzędów funkcji: O, Ω, Θ.
4. (1pkt) Jaka jest najmniejsza wartość n, dla której algorytm o złożoności $100n^2$ działa (na tej samej maszynie) szybciej od algorytmu o złożoności 2^n?

5. (1pkt) Dla każdej funkcji $f(n)$ i czasu t w poniższej tabelce, określ największy rozmiar n danych, dla których algorytm wykona obliczenia w czasie t. Zakładamy, że algorytm rozwiązujący problem potrzebuje $f(n)$ mikrosekund dla danych rozmiaru n.

	1 sekunda	1 minuta	1 godzina	1 dzień	1 miesiąc	1 rok	1 wiek
$\log n$							
$\sqrt{n}$							
n							
$n \log n$							
n^2							
n^3							
2^n							
$n!$							

O ile większe zadania można by rozwiązywać na komputerze 1000 razy szybszym (tj. takim, na którym algorytm potrzebowałby $f(n)$ nanosekund dla danych rozmiaru n)?

6. (1 pkt) Skonstruuj program dla maszyny RAM, który dla danej liczby naturalnej n obliczy $n!$.

   Oszacuj złożoność czasową tego programu przy jednorodnym i logarytmicznym kryterium kosztów. Ustal własną miarę "rozmiaru" danych.

7. (1pkt) Napisz w C lub Pascalu funkcję implementującą podany na wykładzie algorytm, który oblicza n-tą liczbę Fibonacciego (modulo stała) w czasie $O(\log n)$.

8. Napisz rekurencyjne funkcje, które dla danego drzewa binarnego T obliczają:
   - (0,5pkt) liczbę wierzchołków w T;
   - (1pkt) maksymalną odległość między wierzchołkami w T.

9. Napisz procedury, które dla danego drzewa binarnych przeszukiwań T:
   - (0,5pkt) wstawiają zadany klucz do T;
   - (1pkt) usuwają zadany wierzchołek z T;
   - (0,5pkt) dla danego klucza k znajdują następny co do wielkości klucz w drzewie.

10. (1pkt) Napisz funkcję, która dla danej, uporządkowanej rosnąco, tablicy liczbowej T oraz liczby k, obliczy liczbę elementów w T mniejszych od k.

11. Określ z dokładnością do Θ złożoność (przy kryterium jednorodnym) poniższych fragmentów programów:

```
for i ← 1 to n do
    j ← i
    while j < n do
        sum ← P(i, j)
        j ← j + 1
```

```
for i ← 1 to n do
    j ← i
    while j < n do
        sum ← P(i, j)
        j ← j + j
```

    Rozważ dwa przypadki:
    - koszt wykonania procedury $P(i,j)$ wynosi $\Theta(1)$
    - koszt wykonania procedury $P(i,j)$ wynosi $\Theta(j)$

12. (2pkt) Ułóż algorytm dla następującego problemu:

Problem.[1]

dane: $n, m \in \mathcal{N}$

wynik: wartość współczynnika przy x^2 (wzięta modulo m) wielomianu $\underbrace{(...((x-2)^2-2)^2...-2)^2}_{n \text{ razy}}$

Czy widzisz zastosowanie metody użytej w szybkim algorytmie obliczania n-tej liczby Fibonacciego do rozwiązania tego problemu?

Krzysztof Loryś

[1] Zadanie zaczerpnięte ze Sparingu w Programowaniu Zespołowym - Poznań 22.01.2005