Algorytmy i struktury danych (2022/2023)

Ćwiczenia 1

Zadanie 1.1

Dane: dodatnia liczba całkowita n

a[1..n] - tablica liczb całkowitych

Wynik: s* = max({0} ∪ {$\sum_{k=i}^{j} a[k]$: 1 ≤ i ≤ j ≤ n})

a) Oto algorytm obliczania s* wprost z definicji

Algorytm A

```
begin
  s* := 0;
  for i ∈ [1..n] do
    for j ∈ [i..n] do
    begin
      s := 0;
      for k ∈ [i..j] do
        s := s + a[k];        {operacja dominująca}
      s* := MAX(s*,s)
    end
end
```

- Ile **dokładnie** operacji dominujących zostanie wykonanych w algorytmie A?
- Jak długo będzie trwało wykonanie tego algorytmu dla n = 1000^2 przy założeniu, że w 1 sekundzie jest wykonywanych 1000^3 dodawań (operacji dominujących)?

b) Proste usprawnienie Algorytmu A

Algorytm B

```
begin
   s* := 0;
   for i ∈ [1..n] do
   begin
      s := 0;
      for j ∈ [i..n] do
      begin
         s := s + a[j];       {operacja dominująca}
         s* := MAX(s*,s)
      end
end
```

- Na czym polega usprawnienie?
- Ile **dokładnie** operacji dominujących zostanie wykonanych w algorytmie B?
- Jak długo będzie trwało wykonanie tego algorytmu dla n = 1000^2 przy założeniu, że w 1 sekundzie jest wykonywanych 1000^3 dodawań (operacji dominujących)?

c) Szybki algorytm obliczania s*

Algorytm C

```
begin
  s* := 0; p := 0;
  for i ∈ [1..n] do
  begin
    p := p + a[i];      {operacja dominująca}
    s* := MAX(s*,p);
    if p < 0 then
      p := 0
  end
```

- Udowodnij poprawność Algorytmu C podając stosowny niezmiennik pętli „for”.

Zadanie 1.2

Liczby Fibonacciego definiujemy następująco:

$$F_n = \begin{cases} n & n = 0,1 \\ F_{n-1} + F_{n-2} & n > 1 \end{cases}$$

a) Oblicz ile dodawań jest wykonywanych przy liczeniu F_n rekurencyjnie.

b) Zaprojektuj algorytm obliczania liczby F_n wykonujący O(log n) operacji arytmetycznych, z wykorzystaniem wzoru rekurencyjnego:
dla n > 1, $F_{2n-1} = F_n^2 + F_{n-1}^2$ oraz $F_{2n} = F_n^2 + 2F_nF_{n-1}$.

c) Zaprojektuj algorytm obliczania liczby F_n wykonujący O(log n) operacji arytmetycznych, z wykorzystaniem wzoru rekurencyjnego:
dla n > 1, $\begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_{n-1} \\ F_{n-2} \end{bmatrix}$.

Zadanie 1.3

Zaproponuj iteracyjny algorytm, który w czasie O(log n) oblicza $\lfloor \sqrt{n} \rfloor$ dla danej nieujemnej liczby całkowitej n.
Udowodnij poprawność swojego algorytmu.

Zadanie 1.4

Rozważmy następujący algorytm:

Algorytm ?

```
begin
{ x ≤ 100 – liczba całkowita}
  y := x; z := 1;
  while (y ≤ 100) or (z ≠ 1) do
    if y ≤ 100 then
    begin y := y+11; z := z+1 end
    else
    begin y := y – 10; z := z-1 end
end
```

Udowodnij, że Algorytm ? ma własność stopu.

Zadanie 1.5 (opcjonalnie)
Zaprojektuj wydajny algorytm zgodny z poniższą specyfikacją. Uzasadnij poprawność swojego rozwiązania i dokonaj analizy złożoności obliczeniowej swojego algorytmu.

Dane: dodatnia liczba całkowita n > 2

a[1..n] – tablica liczb całkowitych, w której co najmniej 1 element występuje więcej niż $\frac{1}{3}n$ razy (taki element nazywamy słabym przywódcą)

Wynik: e – słaby przywódca w tablicy a

Zadanie 1.6 (opcjonalnie)
W tym zadaniu analizujemy algorytm mnożenia dwóch liczb całkowitych nieujemnych w modelu bitowym – operacjami dominującymi są operacje na bitach.
Dane: n – dodatnia liczba całkowita, $n = 2^k$ dla pewnego $k \geq 0$
x, y - nieujemne liczby całkowite, których zapisy binarne mają długość n
Wynik: iloczyn z = xy
Algorytm 1
```
Iloczyn1(x,y,n)::
begin
  if n = 1 then
    return x*y
  begin
    let x = a*2^(n/2) + b i y = c*2^(n/2) + d ;
    return Iloczyn1(a,c,n/2)*2^n + (Iloczyn1(a,d,n/2)+Iloczyn1(b,c,n/2))*2^(n/2)
           + Iloczyn1(b,d,n/2)
  end
end
```

Uwaga: dodawanie dwóch liczb n bitowych wykonujemy w czasie O(n), podobnie mnożenie przez potęgę dwójki.
a) Podaj równanie rekurencyjne na koszt mnożenia liczb x, y i rozwiąż je.
b) Niech u = (a+b)*(c+d), v = a*c, w = b*d. Wyraź x*y jako wartość wyrażenia zawierającego u, v i w oraz operacje dodawania, odejmowania i mnożenia przez potęgę dwójki. Podaj równanie rekurencyjne na koszt mnożenie x, y w tym przypadku i rozwiąż je. Zastanów się co zrobić w przypadku, gdy liczby (a+b) i (c+d) mają w zapisie binarnym n/2 + 1 bitów.

Zadanie 1.7

Niech n będzie dodatnią liczbą całkowitą i niech S będzie multizbiorem n liczb całkowitych dodatnich. W tym zadaniu za koszt sumowania dwóch liczb przyjmujemy wartość ich sumy.

Rozważmy następujący algorytm:

```
Suma(S)::
begin
   z := 0;
   while |S| ≠ 1 do
   begin
        (x,y) := Para(S);
        S := S \ {x,y};
        S := S ∪ {x+y}
   end;
   return z ∈ S
end;
```

a) W wyniku wywołania funkcji Para otrzymujemy parę elementów z S. Zaimplementuj funkcję Para w taki sposób, żeby koszt obliczania z był jak najmniejszy. Udowodnij poprawność swojego rozwiązania.

b) Załóżmy teraz, że S jest ciągiem i Para zwraca i usuwa dwa sąsiednie elementy w ciągu, a ich sumę wstawia w miejsce tych usuniętych. Zaprojektuj wydajny algorytm, który wyznaczy strategię pobierania par z S tak, żeby w wyniku koszt obliczania z był jak najmniejszy.

Zadanie 1.8

W tym zadaniu rozważamy algorytm sortowania przez wstawianie dla tablicy a[1..n], $n > 0$, zawierającej permutację liczb 1, 2, …, n. Dokonaj analizy pesymistycznej złożoności obliczeniowej tego algorytmu dla następujących przypadków:

a) $|a[i]-a[j]|<2020$, dla każdej pary $1 \le i, j \le n$ takiej, że $|i - j| < 2020$

b) $|i - a[i]| < 2020$, dla każdego $1 \le i \le n$

a) dla co najwyżej 2020 elementów zachodzi $i \neq a[i]$, $1 \le i \le n$