

Zad 3

Procedure bubblesort ($A[1 \dots n]$)

for $i = 1$ to $n-1$:

 for $j = 1$ to $n-i$:

 if ($A[j] > A[j+1]$)

$tmp = A[j]$

$A[j] = A[j+1]$

$A[j+1] = tmp$

Idea: W j -tej iteracji wewnętrznej pętli porównujemy dwie sąsiednie liczby: w razie konieczności je zamieniamy. Po ostatniej iteracji na pozycji $n-i+1$ znajduje się i -ta największa liczba.

Czasowość czasowa: zawsze $\Theta(n^2)$ Uzasadnienie: Wykonanie instrukcji if zajmuje czas stał. Zewnętrzna pętla wykona się $\Theta(n)$ razy, a w każdej jej iteracji wykona się wewnętrzna pętla, $n-i$ razy. Czyli kod wewnętrzny obu pętli wykona się $(n-1) + (n-2) + \dots + 2 + 1$ razy, co się równe $(n-1) \cdot n \cdot \frac{1}{2} = \Theta(n^2)$

Czasowość pamięciowa: $\Theta(1)$ Uzasadnienie: korzystamy tylko z jednej zmiennej pomocniczej tmp.

Zad 4 Mnożenie „po rosyjsku”

$$a_1 = a, a_K = 1, a_{i+1} = \left\lfloor \frac{a_i}{2} \right\rfloor$$

$$b_n = b, b_{i+1} = 2 \cdot b_i \quad (b_i = 2^i b)$$

liczymy $\sum_{i=1}^K b_i$
 a_i nieparzyste

$$\sum_{i=1}^K b_i = \sum_{i=1}^K 2^{i-1} b = b \cdot \sum_{i=1}^K 2^{i-1} = a \cdot b$$

a_i nieparzyste $a_i - np.$

binaryny zapis liczby a

Kryterium jednorodne: U każdego kroku dzielimy a_i na 2, więc wykonujemy $\log_2 n$ kroków. Mnożąc $b_{i+1} = 2 \cdot b_i$ także wykonujemy $\log_2 n$ kroków; sumując $\sum_{i=1}^K b_i$ również. Stąd złożoność czasowa to $O(\log_2 n)$. Złożoność pamięciowa wynosi $O(\log_2 ab)$, bo musimy pamiętać a_i dla $i=1.. \log_2 n$.

Kryterium logarytmiczne: Złożoność czasowa to $O(\log_2 n \cdot \log_2 nm)$, gdzie $m=b$ ponieważ w najgorszym przypadku suma długości operandów w zapisie binarnym to $O(\log_2 ab) = O(\log_2 nm)$. Stąd też złożoność pamięciowa to $O(\log_2 nm)$.

mnozenie_rossyjskie(a,b):

wynik=0

while(a > 0)

if(a % 2 == 0)

wynik += b

a /= 2

b *= 2

return wynik

Zad 5

Chcemy wyznaczyć taka macierz A , że:

$$A \cdot \begin{bmatrix} a_i \\ a_{i-1} \\ \vdots \\ a_{i-k} \end{bmatrix} = \begin{bmatrix} a_{i+1} \\ a_i \\ \vdots \\ a_{i-k+1} \end{bmatrix}$$

jeżeli $a_{i+1} = \alpha_0 a_i + \alpha_1 a_{i-1} + \dots + \alpha_k a_{i-k}$, to

$$A = \begin{bmatrix} \alpha_0 & \alpha_1 & \cdots & \alpha_{i-k+1} & \alpha_{i-k} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & i & 0 \end{bmatrix}$$

Natomiast jeżeli $b_{i+1} = \beta_0 b_i + \beta_1 b_{i-1} + \dots + \beta_k b_{i-k} + \mathcal{U}(n)$, np

$$\mathcal{U}(n) = n^2 + 2n - 5$$

$$B \cdot \begin{bmatrix} b_i \\ b_{i-1} \\ \vdots \\ b_{i-k} \\ n^2 \\ n \\ 1 \end{bmatrix} = \begin{bmatrix} b_{i+1} \\ b_i \\ b_{i-k+1} \\ (n+1)^2 \\ n+1 \\ 1 \end{bmatrix}$$

to

$$B = \begin{bmatrix} \beta_0 & \beta_1 & \cdots & \beta_k & 1 & 2 & -5 \\ 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 2 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 1 \\ \vdots & \ddots & \ddots & \ddots & 0 & 0 & 1 \end{bmatrix}$$

trójkąt Pascala

2ad 6

```
bool s = 0  
for e in A:  
    if(e % 2 == 1)
```

$s = -s$

output s

2) Oznaczenie czasowa: $\Theta(n)$
2) Oznaczenie pamięciowa: $O(1)$

2ad 7

Drewo T będziemy reprezentowali jako tablicę T, gdzie $T[i]$ oznacza wierzchołek - rodnic wierzchołka i .

2) Wtedy chcąc sprawdzić, czy v_i leży na ścieżce z u_i do koneniu będziemy przehodzili od u_i do góry aż natokamy v_i lub zatrzymamy się na koneniu co oznacza, że v_i nie leży na tej drodze.