Wstęp do programowania

Wykład 4. Sortowanie

Plan wykładu

1. Praca z plikami
2. Sortowanie
- sortowanie bąbelkowe
- sortowanie przez wstawianie
- sortowanie przez wybór
- sortowanie stogowe
- sortowanie szybkie
3. Złożoność obliczeniowa

Pliki

```
#include<stdio.h>
#include<stdlib.h>

void main(){
  FILE *f;
  f = fopen("plik.txt", "w"); // otwarcie pliku do zapisu
  for(int i=0; i<1000; i++){
    fprintf(f, "%d\n", rand()); // pisanie do pliku
  }
  fclose(f);  // zamkniecie pliku
}
```

Funkcja `rand()` zwraca liczbę losową (naturalną) z przedziału `[0, RAND_MAX)`.
Najczęściej używane tryby otwarcia pliku: `r`, `w`, `w+`.

Pliki

```
#include<stdio.h>
#include<stdlib.h>

void main(){
  FILE *f;
  int x;
  f = fopen("plik.txt", "r");
  for(int i=0; i<1000; i++){
    fscanf(f, "%d", &x);
    printf("%d\n", x);
  }
  fclose(f);
}
```

Podstawowe operacje na plikach: `fopen()`, `fclose()`, `fprintf()`, `fscanf()`.

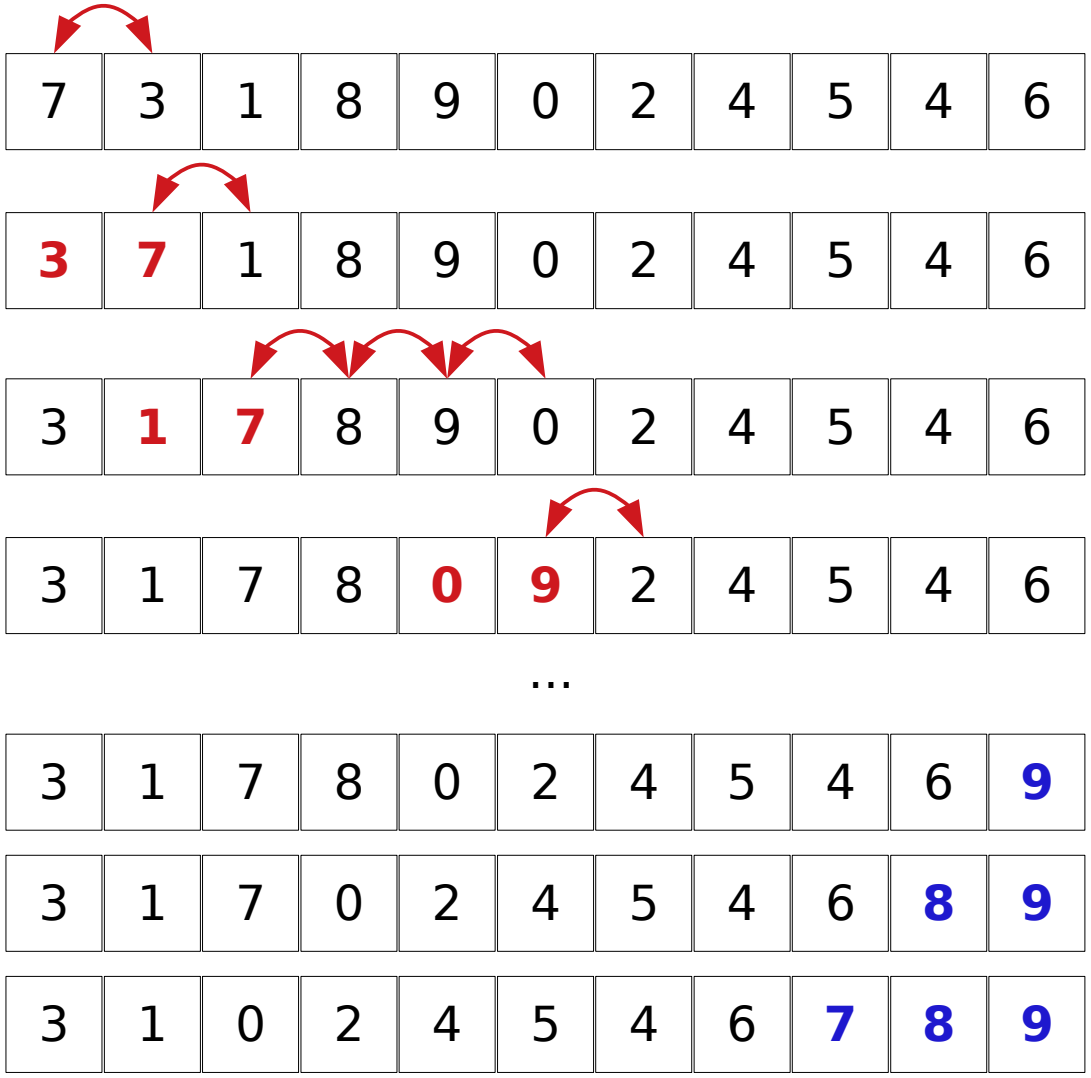
Sortowanie

7	3	1	8	9	0	2	4	5	4	6

0	1	2	3	4	4	5	6	7	8	9

Sortowanie to jedno z podstawowych zagadnień algorytmicznych, które dodatkowo jest powszechnie wykorzystywane w różnego rodzaju oprogramowania. Niewłaściwy dobór metody sortowania może istotnie obniżyć wydajność aplikacji.

Sortowanie bąbelkowe



1. Zacznij od lewej strony
2. Porównaj sąsiednie elementy. Jeśli trzeba to je zamień.
3. Przesuń się o jedną pozycję w prawo.
4. Jeśli doszedłeś do końca tablicy wróć na początek i przejdź do punktu 2.
5. Pętle 2-4 wykonaj tyle razy ile jest elementów w tablicy.

Sortowanie bąbelkowe

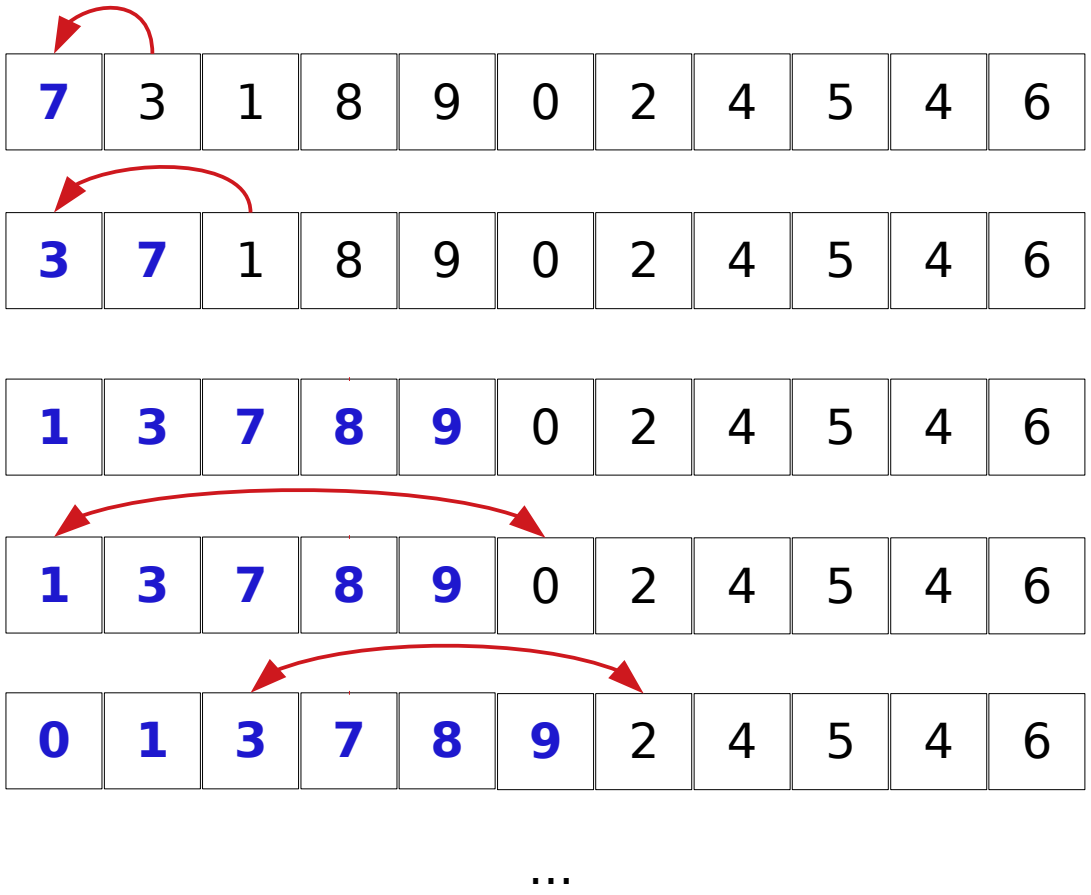
```
void bubblesort(int table[], int size){
  int i, j, temp;
  for (i = 0; i<size-1; i++){
    for (j=0; j<size-1-i; j++){
      if (table[j] > table[j+1]){
        temp = table[j+1];
        table[j+1] = table[j];
        table[j] = temp;
      }
    }
  }
}
```

Źródło: https://pl.wikibooks.org/wiki/Kody_%C5%BAr%C3%B3d%C5%82owe/Sortowanie_b%C4%85belkowe

Ilość operacji: pierwsza pętle `(size-1)`, druga pętla, średnio `(size-1)/2`.

Razem około `(size-1)*(size-1)/2 = 0.5*size`^{2} `+` … `= O(size`^{2}`)`

Sortowanie przez wstawianie



1. Zacznij od lewej strony, która zawiera posortowane elementy (na początku 1 element).
2. Weź pierwszy element z nieposortowanej części i wstaw go na właściwe miejsce w części posortowanej.
3. Powtarzaj tą czynność dopóki cała tablica nie będzie posortowana.

Sortowanie przez wstawianie

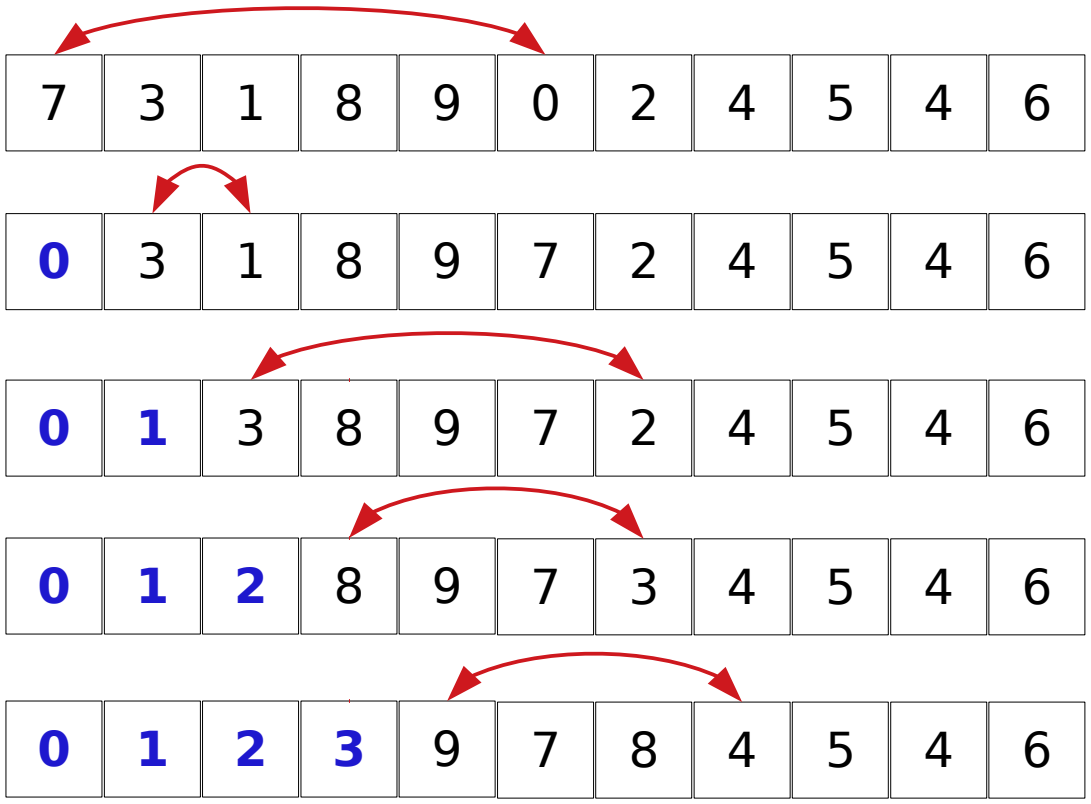
```
void insertSort(int a[], int size){
  int i, j, value;
  for (i = 1; i < size; ++i){
    value = a[i];
    for (j = i - 1; j >= 0 && a[j] > value; --j){
      a[j + 1] = a[j];
    }
    a[j + 1] = value;
  }
}
```

Źródło: `https://pl.wikibooks.org/wiki/Kody_%C5%BAr%C3%B3d%C5%82owe/Sortowanie_przez_wstawianie`

Ilość operacji: pierwsza pętle `(size-1)`, druga pętla, średnio `(size-1)/2`.

Razem około `(size-1)*(size-1)/2 = 0.5*size`^{2} `+ ... = O(size`^{2}`)`

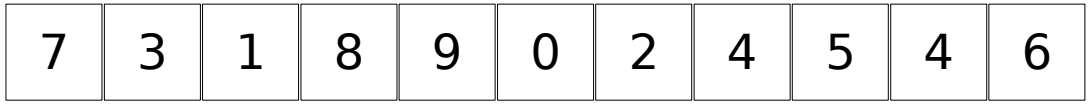
Sortowanie przez wybieranie

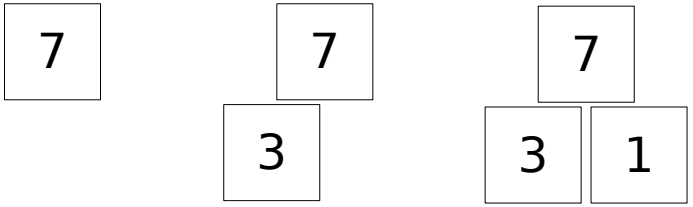


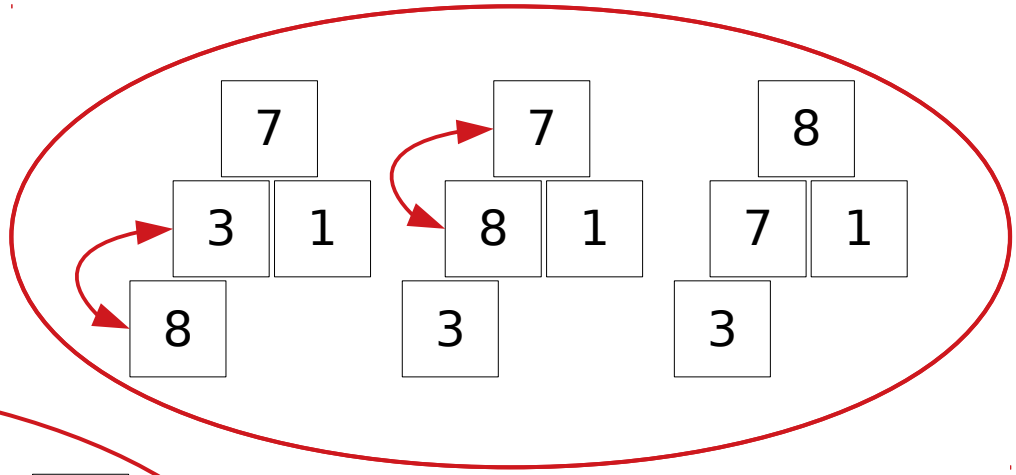
...

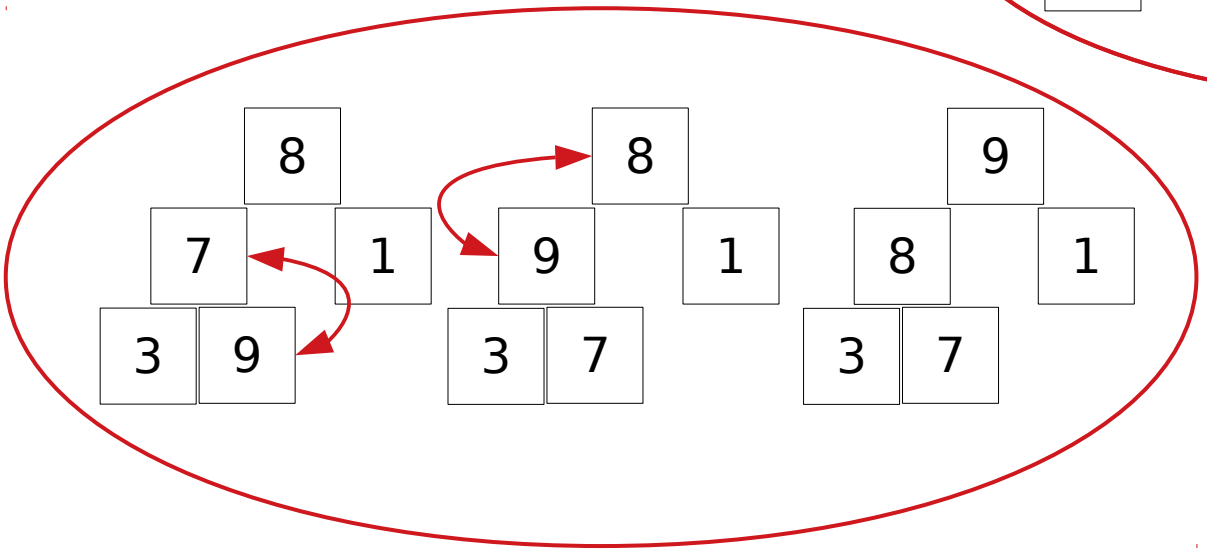
1. Zacznij od lewej strony, która zawiera posortowane elementy (na początku 0 elementów).
2. Znajdujemy najmniejszy element w nieposortowanej części i zamieniamy go z pierwszym elementem nieposortowanej części.
3. Powtarzaj tą czynność dopóki cała tablica nie będzie posortowana.

Sortowanie stogowe



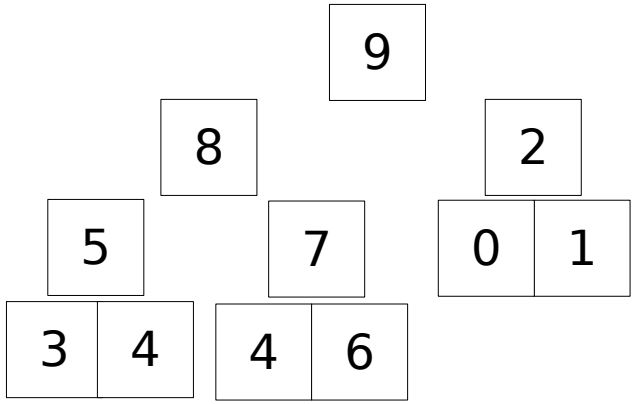


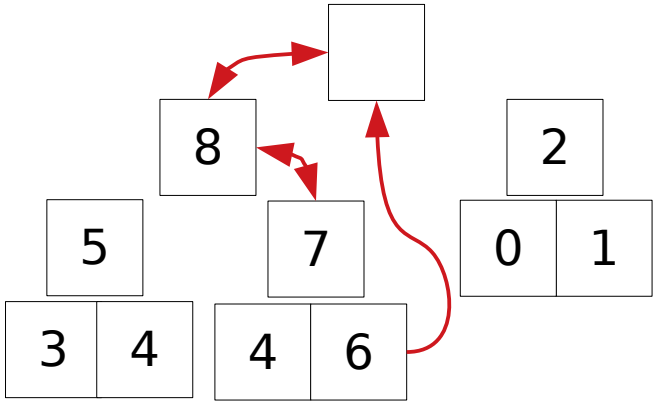




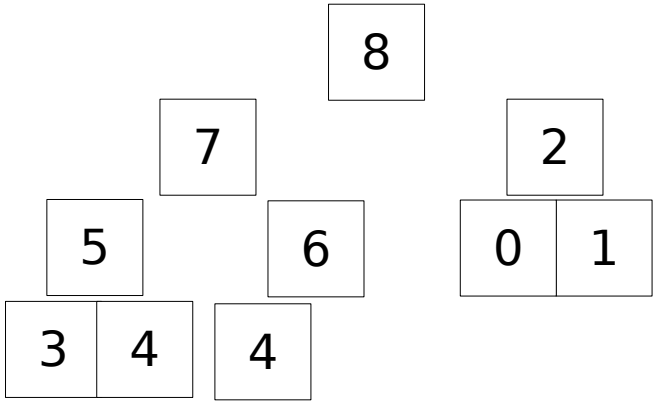
Budowa stogu – drzewo binarne z warunkiem, że „dzieci” mają wartość mniejszą niż „rodzice”.

Sortowanie stogowe



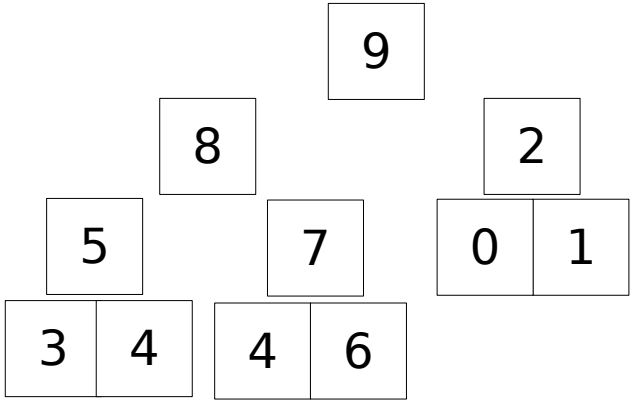


Rozbiór stogu – zdejmujemy element ze szczytu, przekładamy tam element z dołu a następnie przywracamy warunek stogu.



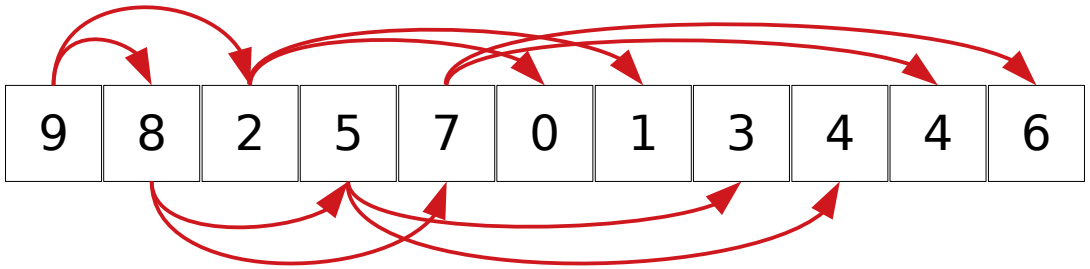
Sortowanie stogowe

Stóg może być zapisany w tablicy!



Dzieci elementu a[i] są w komórkach a[2*i+1] oraz a[2*i+2].

Pamiętamy, że indeks i zaczyna się od 0.



Sortowanie stogowe

Złożoność obliczeniowa:

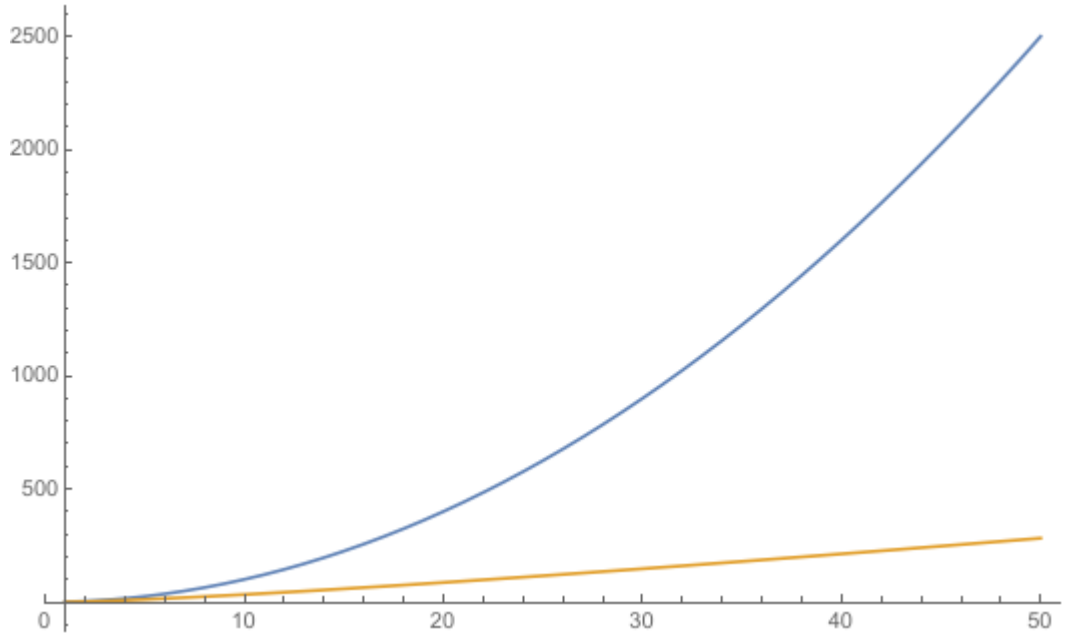
Wysokość kopca – około $\log_2(size)$

faza budowy kopca:
$size*\log_2(size)$

faza rozbioru kopca:
$2*size*\log_2(size)$ – bo dw porównania (z dwójką „dzieci”)

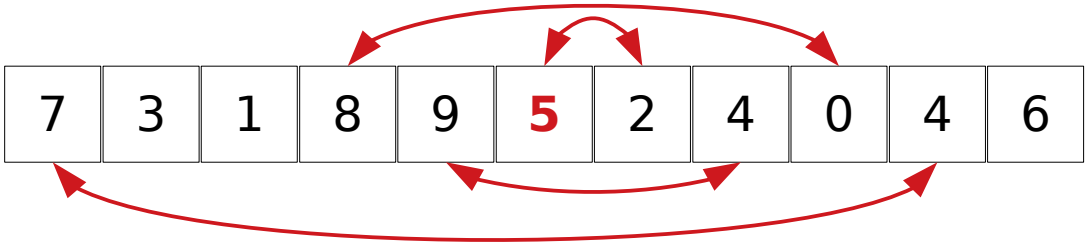
Razem: $3*size*\log_2(size) = O(size*\log_2(size))$

Sortowanie stogowe

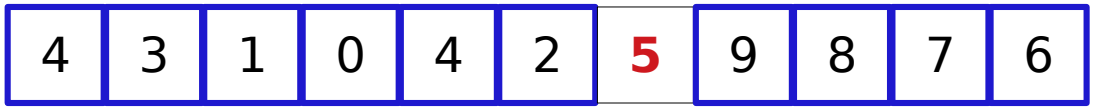


`O(n^2) vs O(n*log(n))`

Sortowanie szybkie



1. Wybieramy element rozdzielający wewnątrz tablicy.
2. Idziemy od lewej strony szukając elementu większego od rozdzielającego i od prawej strony szukając większego. Zamieniamy te elementy i kontynuujemy wędrówkę do momentu spotkania.



Stosujemy powyższą procedurę rekurencyjnie do tablicy po lewej i po prawej stronie elementu rozdzielającego.

Sortowanie szybkie

```
void qsort(void *base, size_t nmemb, size_t size,
           int (*compar)(const void *, const void *));
```

base – tablica do posortowania

nmemb – ilość elementów w tablicy

size - rozmiar pojedynczego elementu, np. **sizeof(int)**

compar – wskaźnik do funkcji porównującej elementy

Sortowanie

Dziękuję za uwagę