# Heapsort

## Условие

Добре известен алгоритъм наречен heapsort е детерминистичен сортиращ алгоритъм със сложност O(nlogn) като време и O(1) като допълнителна памет. Нека опишем възходяща сортировка на масив от различни цели числа.

Алгоритъмът работи в две фази. В първата фаза масивът се превръща в пирамида. Масив a[1...n] от цели числ се нарича пирамида ако за всяко $1 <= i <= n$ важат следните условия:

ако $2i < n + 1$ тогава a[i] > a[2i];

ако $2i+1 < n + 1$ тогава a[i] > a[2i + 1].

Можем да интерпретираме масива като двоично дърво, смятайки че a[2i] и a[2i + 1] са наследници на a[i]. В терминологията на дървото свойствата на пирамидата означават че стойността на всеки връх е по-голяма от тази на наследниците му.

Във втората част на алгоритъма, пирамидата се превръща в сортиран масив. От свойствата на пирамидата най-големия елемент в масива-пирамида е a[1]. Нека го сменим с a[n], сега най-големия елемент в масива е на правилното си място в сортирания масив. Тази операция се нарича извличане на максимума.

Сега нека разгледаме частта от масива a[1...n - 1]. Тя може да не е пирамида, защото условието за пирамида може да се провали за $i = 1$. Ако е така (тоест или a[2], или a[3], или и двете са по-големи от a[1]) нека сменима най-голямият наследник на a[1] с него, възтановявайки свойтвото за $i = 1$. Сега е възможно условието за пирамида да се провали за позицията която заема старата стойност на a[1]. Приложете същата процедура към нея, сменяйки я с по-голямият и наследник.Продължавайки по този начин ще превърнем целия масив a[1...n - 1] в пирамида. Процедурата се нарича пресяване надолу. След като превърнем a[1...n - 1] в масив чрез пресяване извличаме максимума отново, поставяйки втория по големина елемент в a[n - 1], и т.н.

Например, нека разгледаме как масива a = (5, 4, 2, 1, 3) се превръща в сортиран масив. Нека направим първото извличане на максимума. Масива се превръща в (3, 4, 2, 1, 5). Условието за пирамида се разваля от a[1] = 3 защото наследника му a[2] = 4 е по-голям от него. Правим пресяване надолу сменяйки a[1] и a[2]. Сега имаме масива ( 4, 3, 2, 1, 5). Изпълнени са условията за пирамида и пресяването приключва. Отново извлияаме максимума. Новия масив е (1, 3, 2, 4, 5). Отново условието за пирамида се проваля за a[1]; като пресеем получаваме (3, 1, 2, 4, 5) което е пирамида. Пак извличаме максимума и получаваме (2, 1, 3, 4,

5). Този път условията са изпълнени за всички елементи, затова правим извличане на максимума, получавайки (1, 2, 3, 4, 5). Началото на масива е пирамида, и с последното извличане на минимум получаваме (1, 2, 3, 4, 5).
Знаем че превръщането на произволен масив в пирамида става със сложност O(n). Затова, операцията която поглъща най-много време е пресяването $O(n \log n)$.
В тази задача трябва да намерите масив-пирамида съдържащ числата от 1 до n, такъв че когато го превърнем в сортиран масив, най-големия брой размени във всички пресявания е максималния възможен. В горния пример номер на размените е 1 + 1 + 0 + 0 + 0 = 2, което не е максималното. (5, 4, 3, 2, 1) дава максимален брой 4 размени за $N = 5$.

**Вход**
Използва се стандартния вход. Съдържа числото $n$ $(0 < n < 50001)$.

**Изход**
Използва се стандартния изход. Изведете масив съдържащ n различни числа от 1 до n, такъв че да е пирамида, и при обръщането му в сортиран масив, броя нна размените е възможно най-голям. Отделете числата със интервали.

Примерен Вход
```
6
```

Примерен Изход
```
6 5 3 2 4 1
```

**Ограничения:**
Време: 0.25s
Памет: 128MB

**Заглавен коментар (header)**

| за C: | за C++: | за Pascal: |
|---|---|---|
| /* | /* | { |
| TASK:heapsort | TASK:heapsort | TASK:heapsort |
| LANG:C | LANG:C++ | LANG:Pascal |
| */ | */ | } |