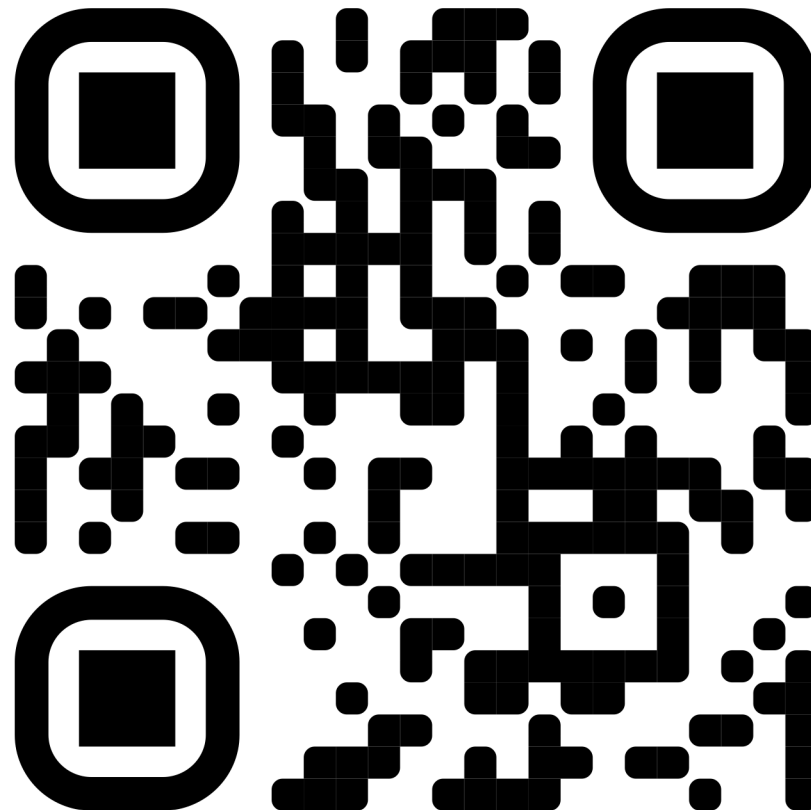


Сортировки

u.to/72D_Gg

Лекция 8, 26 марта, 2021



Лектор:

Дмитрий Северов, кафедра информатики 608 КПМ

dseverov@mail.mipt.ru

Обратная связь : **u.to/7Wn7Gg**

Сортировки

- Введение
- Простые: $O(n^2)$
- Сложные: $O(n \cdot \log n)$

Задача сортировки

- Для некоторой последовательности

$$a_1, a_2, \dots, a_n$$

найти перестановку её элементов в таком порядке

$$a'_1, a'_2, \dots, a'_n,$$

что при заданной функции f справедливо отношение:

$$f(a'_1) \leq f(a'_2) \leq \dots \leq f(a'_n)$$

- Функция упорядочивания $f(x)$ вычисляется не при каждом сравнении, а однажды и содержится в каждом элементе в виде явной компоненты – ключа

Типы сортировки

■ Внутренняя

- все элементы находятся в памяти машины

■ Внешняя

- массив данных настолько велик, что в ОЗУ машины может храниться только лишь часть данных

■ Устойчивая

- сохраняет порядок размещения элементов в массиве, содержащем одинаковые ключи

Основные характеристики

- Время – характеристика вызывающая наибольший интерес
- Дополнительный объем оперативной памяти, используемый алгоритмом
 1. Не требуется (in place);
 2. Для размещения указателей или индексов массивов;
 3. Для размещения еще одной копии массива.

Вспомогательные средства

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;

const int N = 10;

unsigned long long start, end;
unsigned long long access_counter(){ asm("rdtsc"); }

void init(int a[],int l,int r) {
    for(int i=l;i<=r;i++) a[i]=rand(); }

void init(double a[],int l,int r){
    double h=2.*M_PI/(r-1);
    for(int i=l;i<=r;i++) a[i]=sin(h*i); }
```

```
template <typename T>
    void print(T a[], int l, int r) {
        for(int i=l;i<=r;i++)
            cout << a[i]<< (i==r? '\n' : ' ');
    }
```

```
template <typename T>
    void exch(T& A, T& B) {
        T t=A; A=B; B=t;
    }
```

```
template <typename T>
    bool compexch(T& A, T& B) {
        if(B<A) { exch(A,B); return true;
        } else return false;
    }
```

Простые сортировки: $O(n^2)$

1. Вставками
2. Выбором
3. Пузырьком
4. Гномами
5. Перемешиванием

Сортировка вставками

```
template <typename T>
void insert(T a[], int l, int r) {
    for(int i=l+1; i<=r; i++)
        for(int j=i; j>l; j--)
            if( !compech(a[j-1], a[j]) )
                break;
}
```



Сортировка выбором

```
template <typename T>
void slct(T a[], int l, int r) {
    for(int i=l;i<r;i++) {
        int min=i;
        for(int j=i+1;j<=r;j++)
            if(a[j]<a[min]) min=j;
        exch(a[i],a[min]); }
}
```



Сортировка гномами [цветочных горшков]

Если нет горшка сзади, то шагает вперёд.

Если горшки впереди и сзади стоят верно,

то шагает вперёд,

иначе меняет горшки местами и шагает назад.

Если нет горшка впереди, то останавливается.

Сортировка гномами

```
template <typename T>
void dwarf(T a[],int l,int r) {
    int i=l;
    while(i<r)
        if(!compech(a[i],a[i+1]) || i==l) i++;
        else i--;
}
```



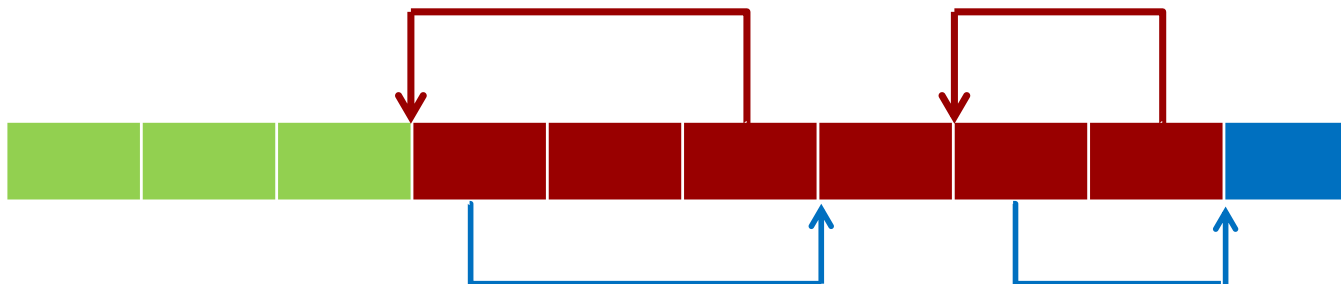
Сортировка пузырьком

```
template <typename T>
void bubble(T a[], int l, int r) {
    int b=1;
    for(int i=l; (i<r) && b; i++){ b=0;
        for(int j=r; j>i; j--)
            b = compech(a[j-1],a[j]) || b;}
}
```



Сортировка перемешиванием

```
template < typename T >
void shaker(T a[], int l, int r) {
    do {
        for(j=k=r; j>l; j--)
            if(compexch(a[j-1], a[j])) k=j;
        l=k;
        for(j=k=l+1; j<=r; j++)
            if(compexch(a[j-1], a[j])) k=j;
        r=k-1;
    }while(l<r);
}
```



Сложные сортировки

■ Свойства

- В среднем $O(n \log n)$
- Возможна деградация до $O(n^{3/2})$ и даже $O(n^2)$
- Не всегда «in line»

■ Шелла

■ Быстрая (Хоара)

■ Слиянием

■ Пирамидальная

Сортировка Шелла (вставка несоседних)

```
template <typename T>
void shell(T a[], int l, int r) {
    int h;
    for(h=1; h<=(r-l)/3; h=3*h+1);
    for(;h>0; h/=3)
        for(int i=l+h; i<=r; i++) {
            int j=i; T v=a[i];
            while( j>=l+h && a[j-h]>v ) {
                a[j]=a[j-h]; j-=h; }
            a[j]=v; }
}
```

1 4 13 40 121 364 1093 3280 9841 ... $\sim O(N^{3/2})$

Сортировка Шелла

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
E	U	H	I	K	G	K	B	M	Y	A	K	B	B	G	▸
B	G	H	I	K	G	K	B	M	Y	A	K	B	E	U	h=13
B	E	A	B	B	G	H	I	K	G	K	K	M	Y	U	h=4
A	B	B	B	E	G	G	H	I	K	K	K	M	U	Y	■

```

for(int i=l+h;i<=r;i++) { int j=i; T v=a[i];
    while( j>=l+h && a[j-h]>v ) { a[j]=a[j-h]; j-=h; }
    a[j]=v; }
    
```

Сортировка Шелла

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
E	U	H	I	K	G	K	B	M	Y	A	K	B	B	G	▶
B	G	H	I	K	G	K	B	M	Y	A	K	B	E	U	h=13
B	E	A	B	B	G	H	I	K	G	K	K	M	Y	U	h=4
A	B	B	B	E	G	G	H	I	K	K	K	M	U	Y	■

$4^{i+1} + 3 \cdot 2^{i+1} : 1 \ 8 \ 23 \ 77 \ 281 \ 1073 \ 4193 \dots O(N^{4/3})$

$2^i : 1 \ 2 \ 4 \ 16 \ 32 \ 64 \ 128 \dots O(N^2)$

Быстрая сортировка (Хоара)

```
template <typename T>
void quicksort(T a[],int l,int r) {
    int i=l-1,j=r; T x=a[r];
    for(;;) {
        while(a[++i]<x);
        while(x<a[--j]) if(j==l) break;
        if(i>=j)break;
        exch(a[i],a[j]);
    }
    exch(a[i],a[r]);
    if(l<i-1) quicksort(a,l,i-1);
    if(r>i+1) quicksort(a,i+1,r);
}
```

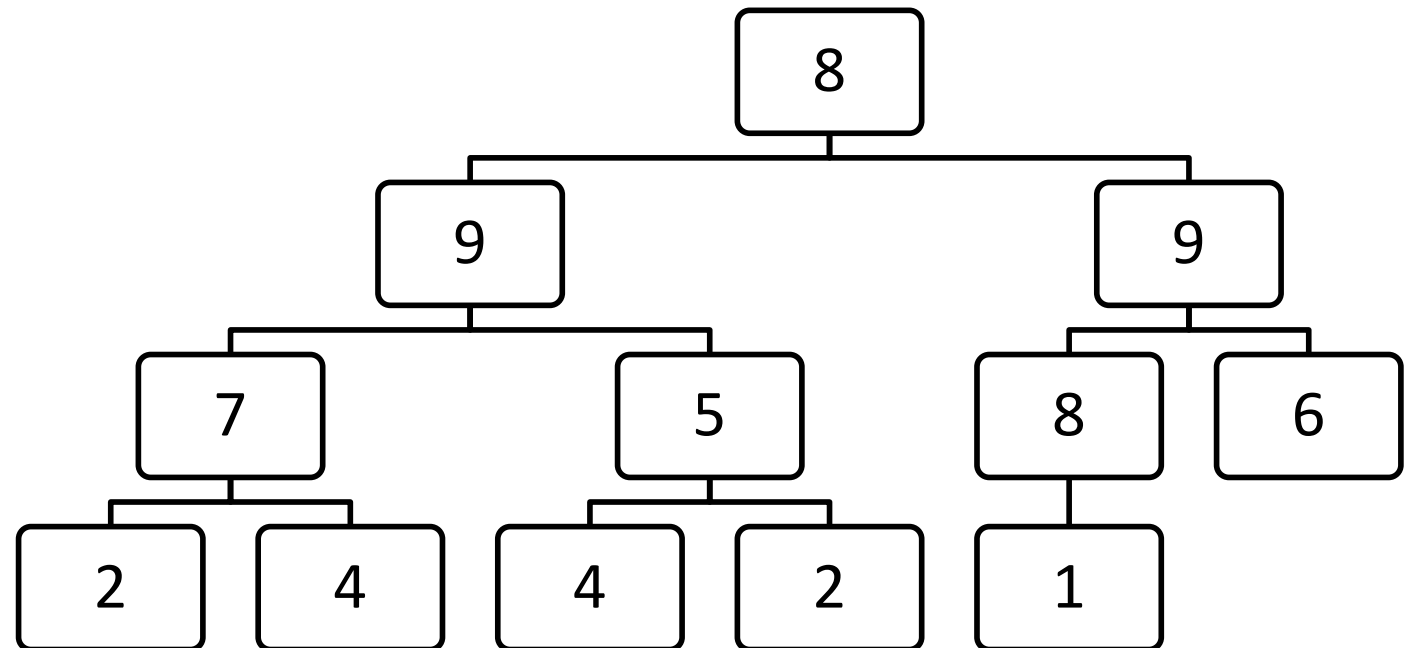
Сортировка слиянием

```
template <typename T>
void merge(T a[], int l, int m, int r) {
    int i, j; static T aux[N];
    for(i=m+1; i>l; i--) aux[i-1]=a[i-1];
    for(j=m; j<r; j++) aux[r+m-j]=a[j+1];
    for(int k=l; k<=r; k++)
        a[k]=aux[j]<aux[i]?aux[j--]:aux[i++];
}
```

```
template <typename T>
void mergesort(T a[], int l, int r) {
    if(r<=l) return; int m=(r+l)/2;
    mergesort(a, l, m); mergesort(a, m+1, r);
    merge(a, l, m, r);
}
```

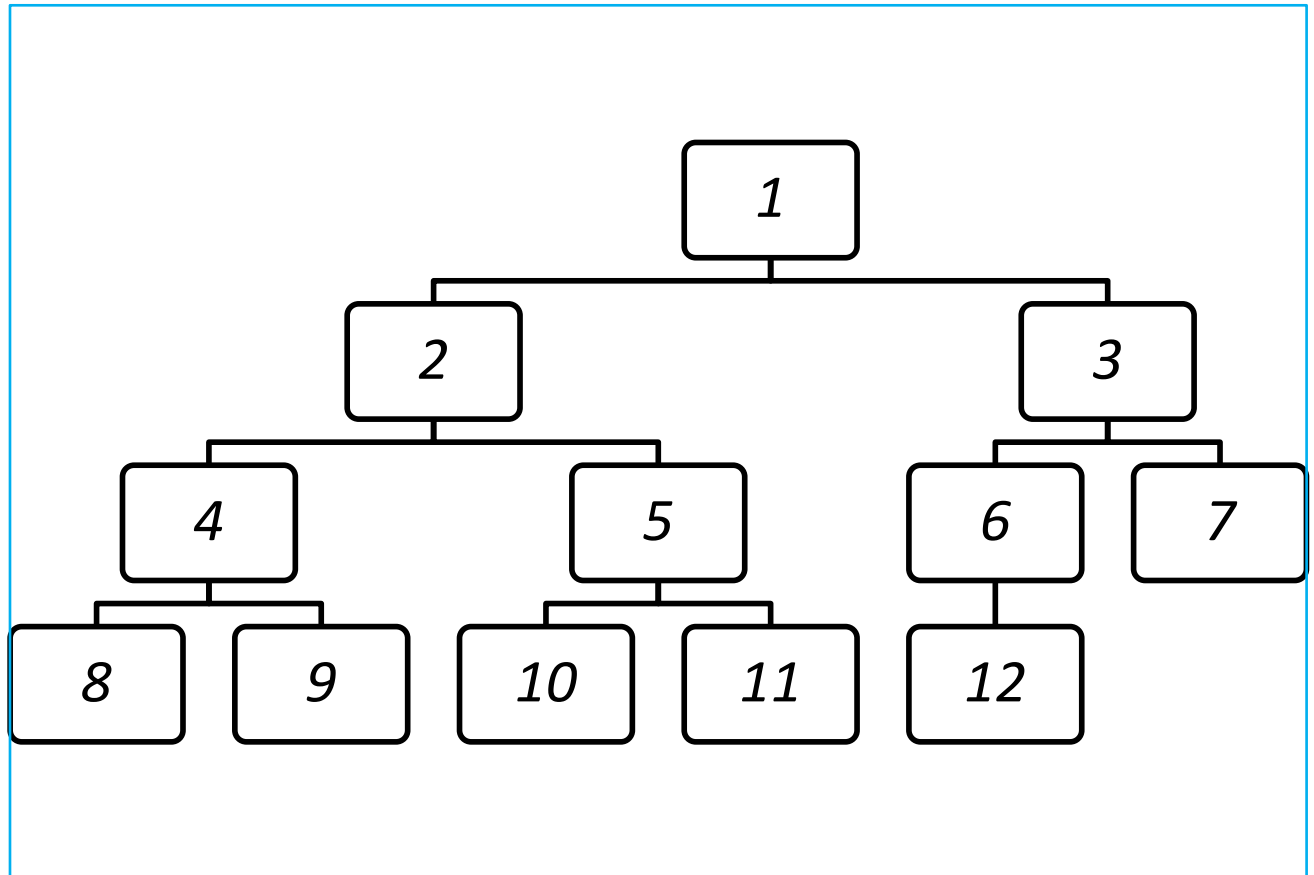
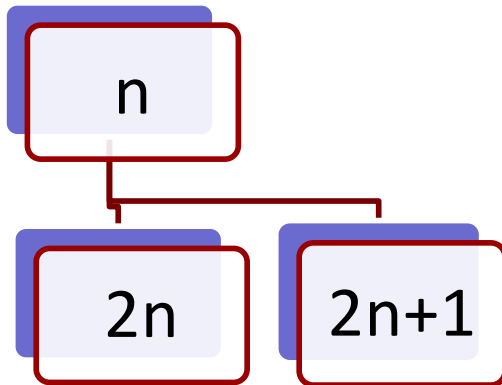
Бинарное сортирующее дерево (куча)

1. Ключ узла не меньше ключей справа и слева.
2. Разница глубин листьев не более 1
3. Последний слой заполняется слева направо без пустот



Структура данных кучи

■ Индексы узлов в массиве



1	2	3	4	5	6	7	8	9	10	11	12
---	---	---	---	---	---	---	---	---	----	----	----

Действия с кучей

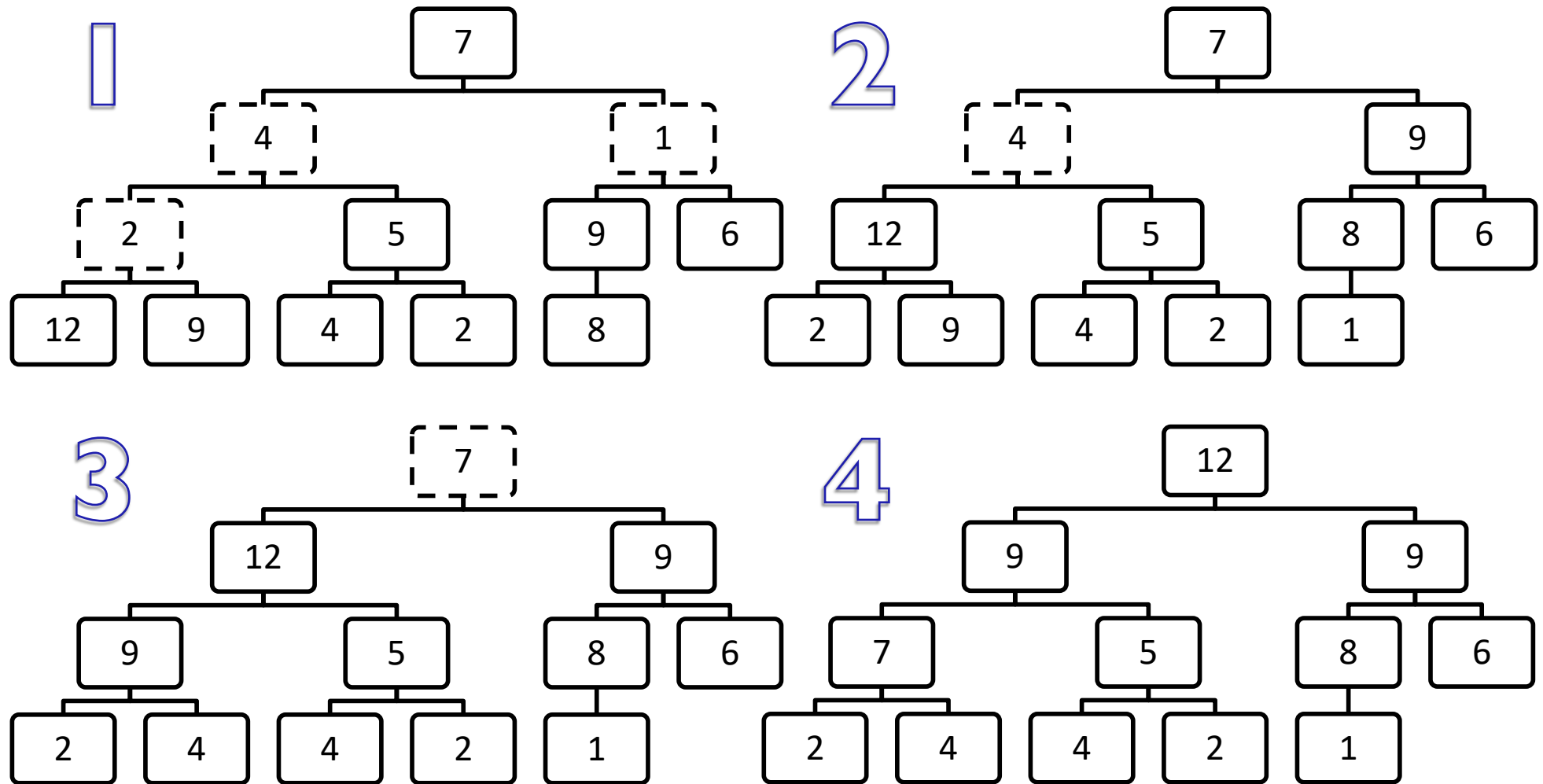
- Точечное исправление кучи: $O(\log n)$
 - По необходимости поменять узел с потомком и исправить ниже
- Построение кучи: $O(n)$
 - Исправление массива до кучи справа налево
- Удаление максимального: $O(\log n)$
 - Поменять первый с последним, укоротить, исправить
- Пирамидальная сортировка : $O(n \log n)$
 - Удалить все максимальные по очереди

Пирамидальная сортировка

```
template <typename T>
void fixDown(T a[],int k,int N) { int j;
    while((j=2*k)<=N) {
        if(j<N && a[j]<a[j+1]) j++;
        if(!compech(a[j],a[k])) break;
        k=j; } }

template <typename T>
void heapsort(T a[],int l,int r) {
    int N=r-l+1; T* pq=a+l-1;
    for(int k=N/2;k>=1;k--) fixDown(pq,k,N);
    while(N>1) { exch(a[0], pq[N]);
        fixDown(pq,1,--N); } }
```


Построение кучи



Исходный массив

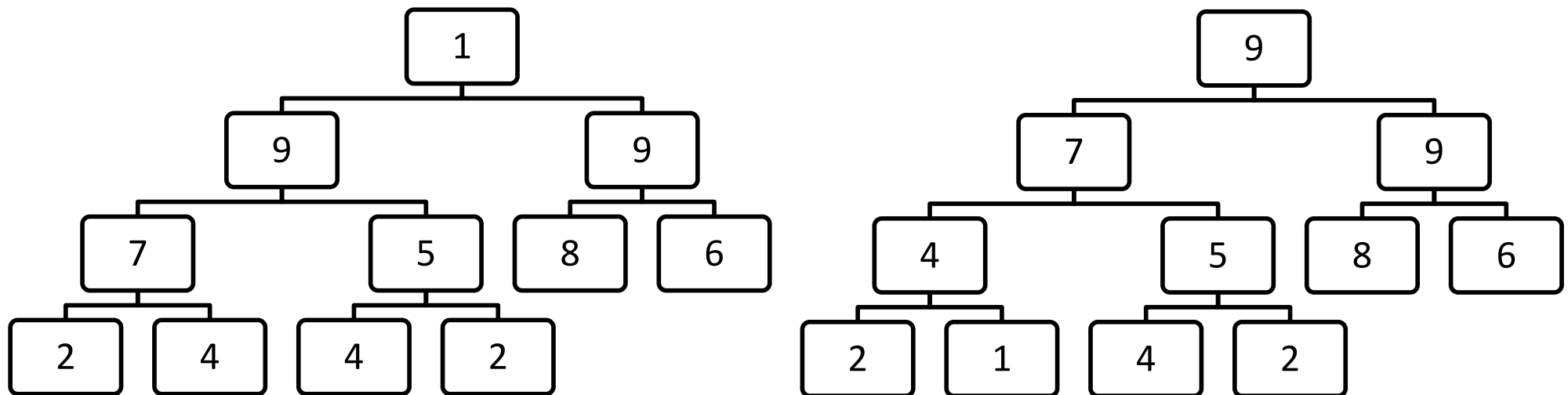
7	4	1	2	5	9	6	12	9	4	2	8
---	---	---	---	---	---	---	----	---	---	---	---

Исходная куча

12	9	9	7	5	8	6	2	4	4	2	1
----	---	---	---	---	---	---	---	---	---	---	---

2 массив

1	9	9	7	5	8	6	2	4	4	2	12
---	---	---	---	---	---	---	---	---	---	---	----



2 куча

9	7	9	4	5	8	6	2	1	4	2	12
---	---	---	---	---	---	---	---	---	---	---	----

```

#include <iostream>
#include <cmath>
#include <iomanip>
#include <stdlib.h>
using namespace std;
const int N = 1000;
unsigned long long start, end;
unsigned long long access_counter() { asm("rdtsc"); }
void init(int a[],int l,int r) { for(int i=l;i<=r;i++) a[i]=rand(); }
void init(double a[],int l,int r) { double h=2.*M_PI/(r-l); for(int i=l;i<=r;i++) a[i]=sin(h*i); }
template <typename T> void print(T a[], int l, int r) { for(int i=l;i<=r;i++) cout << a[i] << (i==r?'\\n':' '); }

#include "Fun_Sort.cpp"

int main() {
const char* Name[] = {"insert", "slct", "bubble", "dwarf",
                     "shaker", "shell", "quicksort", "mergesort", "heapsort"};
//double *a=new double[N];
//int (*PFcmp)(const void*,const void*) = Fcmp<double>;
//void (*PF[])(double[],int,int) = {insert, slct, bubble, dwarf, shaker, shell, quicksort, mergesort, heapsort};
int *a=new int[N];
int (*PFcmp)(const void*,const void*) = Fcmp<int>;
void (*PF[])(int[],int,int) = {insert, slct, bubble, dwarf, shaker, shell, quicksort, mergesort, heapsort};
#define INT 1

    for(int n=0;n<9;n++) {
        init(a,0,N-1);
    }
}

```

```

    start = access_counter();
    (*PF[n])(a,0,N-1);
    end = access_counter();
    cout << Name[n] << " time was:      " << setw(10) << setfill(' ') << end - start << " \n"; }

init(a,0,N-1);
start = access_counter();
qsort(a,N,sizeof(a[0]),PFcmp);
end = access_counter();
cout << "qsort lib time was:      " << setw(10) << setfill(' ') << end - start << " \n";

cout << "quicksort";
// init(a,0,N-1);
start = access_counter();
quicksort(a,0,N-1);
end = access_counter();
cout << " time was: Sp cs" << setw(10) << setfill(' ') << end - start << " \n";
#if defined INT
    cout << "qsB      ";
    init(a,0,N-1);
    start = access_counter();
    qsB(a,0,N-1,0);
    end = access_counter();
    cout << " time was:      " << setw(10) << setfill(' ') << end - start << " \n";
#endif
return 0;
}

```

```

template <typename T> void print(T a[], int l, int r) { for(int i=l;i<=r;i++) cout << a[i] << (i==r?'\\n':' '); }

template <typename T> inline void exch(T& A,T& B) { T t=A; A=B; B=t; }
template <typename T> inline bool compexch(T& A,T& B) { if(B<A) { exch(A,B); return true; } else return false; }

template <typename T> void insert(T a[], int l, int r) { for(int i=l+1;i<=r;i++) for(int j=i;j>l;j--)
    if(!compexch(a[j-1],a[j])) break; }

template <typename T> void slct(T a[], int l, int r) { for(int i=l;i<r;i++) { int min=i; for(int j=i+1;j<=r;j++)
    if(a[j]<a[min]) min=j; exch(a[i],a[min]); } }

template <typename T> void bubble(T a[], int l, int r) { int b=1; for(int i=l; (i<r) && b; i++)
    { b=0; for(int j=r; j>i; j--) b = compexch(a[j-1],a[j]) || b; } }

template <typename T> void dwarf(T a[], int l, int r) { int i=l; while(i<r) if(!compexch(a[i],a[i+1]) || i==l) i++; else i--; }

template <typename T> void shaker(T a[], int l, int r) { int j,k;
do { for(j=k=r;j>l;j--) if(compexch(a[j-1],a[j])) k=j; l=k;
    for(j=k=l+1;j<=r;j++) if(compexch(a[j-1],a[j])) k=j; r=k-1;}
while(l<r); }

template <typename T> void shell(T a[], int l, int r) { int h, j; for(h=1; h<=(r-l)/3; h=3*h+1);
for(;h>0; h/=3) for(int i=l+h;i<=r;i++) { j=i; T v=a[i]; while( j>=l+h && v<a[j-h] ) { a[j]=a[j-h]; j-=h; } a[j]=v; } }

```

```

template <typename T> void quicksort(T a[], int l, int r) {
int i=l-1,j=r; T x=a[r];
    for(;;) { while(a[++i]<x); while(x<a[--j]) if(j==l) break; if(i>=j)break; exch(a[i],a[j]); }
    exch(a[i],a[r]);
    if(l<i-1) quicksort(a,l,i-1);
    if(r>i+1) quicksort(a,i+1,r);
}

```

```

template <typename T> int Fcmp(const void* a, const void* b) {
T *A = (T*)a, *B = (T*)b;
    if( *A > *B) return 1;
    if( *A < *B) return -1;
    return 0; }

```

```

template <typename T> void merge(T a[], int l, int m, int r) {
int i,j; static T aux[N];
    for(i=m+1;i>l;i--) aux[i-1]=a[i-1];
    for(j=m;j<r;j++) aux[r+m-j]=a[j+1];
    for(int k=l;k<=r;k++) a[k]=aux[j]<aux[i]?aux[j--]:aux[i++]; }

```

```

template <typename T> void mergesort(T a[], int l, int r) {
    if(r<=l) return;    int m=(r+l)/2;
    mergesort(a,l,m); mergesort(a,m+1,r);
    merge(a,l,m,r); }

```

```

template <typename T> void fixDown(T a[],int k,int N) { int j;
    while((j=2*k)<=N) { if(j<N && a[j]<a[j+1]) j++; if(!compexch(a[j],a[k])) break; k=j; } }

template <typename T> void heapsort(T a[],int l,int r)
{ int N=r-l+1; T* pq=a+l-1;
  for(int k=N/2; k>=1; k--) fixDown(a,k,N);
  while(N>1) { exch(a[0],pq[N]); fixDown(pq,1,--N); }
}

const int Bits_num=32;

inline bool digit(int a,int d) { return a & (1<<Bits_num-1-d); }

void qsB(int a[], int l, int r, int d) {
  int i=l, j=r;
  if(l>=r || d>Bits_num-1) return;
  while(i != j) {
    while(!digit(a[i],d) && (i<j) ) i++;
    while(digit(a[j],d) && (j>i) ) j--;
    exch(a[i],a[j]); }
  if(!digit(a[r],d)) j++;
  qsB(a,l,j-1,d+1);
  qsB(a,j,r,d+1);
}

```

insert	time was:	22094121
slct	time was:	5267878
bubble	time was:	68018580
dwarf	time was:	39161367
shaker	time was:	28256494
shell	time was:	408324
quicksort	time was:	383915
mergesort	time was:	1830759
heapsort	time was:	1112706
qsort lib	time was:	613242
quicksort	time was: Sp cs	3281768
qsB	time was:	2088646

Process exited after 0.2338 seconds with return value 0
Для продолжения нажмите любую клавишу . . .

Пример: Сортировка строк

D:\A1\Lect2015\Exmp\7\My_sort.cpp - [Executing] - Dev-C++ 5.10

Файл Правка Поиск Вид Проект Выполнить Сервис AStyle Окно Справка

(globals) Скомпилировать и выполнить (F11) TDM-GCC 4.8.1 32-bit Profiling

My_sort.cpp sort.cpp Fun_Sort.cpp

```
1  #include <iostream>
2  #include <fstream>
3  #include <stdlib.h>
4  #include <iomanip>
5
6
7  using namespace std;
8  const int N=200;
9
10
11 #include "My_str_new.h"
12
13
14 unsigned long long start, end;
15 unsigned long long access_counter() { asm("rdtsc"); }
16
17 template <typename T> void print(T a[], int l, int r) { for(int i=l; i<=r; i++) cout << a[i]; }
18
19
20 #include "Fun_Sort.cpp"
21
22
23 int main(int argn, char** argv) {
24
25     const char* Name[] = {"insert", "slct", "bubble", "dwarf", "shaker",
26                          "shell", "quicksort", "mergesort", "heapsort"};
27
```

Компилятор Ресурсы Журнал компиляции Отладка Результаты поиска

Line: 13 Col: 1 Sel: 0 Lines: 52 Length: 1401 Вставка Done parsing in 0,062 seconds

D:\A1\Lect2015\Exmp\7\My_sort.cpp - [Executing] - Dev-C++ 5.10

Файл Правка Поиск Вид Проект Выполнить Сервис AStyle Окно Справка

(globals)

My_sort.cpp sort.cpp Fun_Sort.cpp

```
27
28 My_string a[N],b[N]; int r=0,n;
29
30 int (*Pfcmp)(const void*, const void*) = Fcmp<My_string>;
31 void (*PF[])(My_string[],int,int) = {insert, slct, bubble, dwarf, shaker, shell, quicksort, mergesort, heapsort};
32
33 ifstream in(argv[1]);
34 if(!in) { cerr << "Cann't open data file:" << argv[1] << endl; return 1;}
35 while(in >> a[r]) { b[r]=a[r]; r++; }
36
37 for(n=0;n<9;n++) {
38     start = access_counter();
39     (*PF[n])(a,0,r-1);
40     end = access_counter();
41     cout << Name[n] << " time was:      " << setw(10) << setfill(' ') << end - start << " \n";
42     for(int i=0;i<r;i++) a[i]=b[i];
43 }
44 start = access_counter();
45 qsort(a,r-1,sizeof(My_string),Pfcmp);
46 end = access_counter();
47 cout << "qsort    time was:      " << setw(10) << setfill(' ') << end - start << " \n";
48 //     for(int i=0;i<r;i++) a[i]=b[i];
49
50 return 0;
51 }
52
```

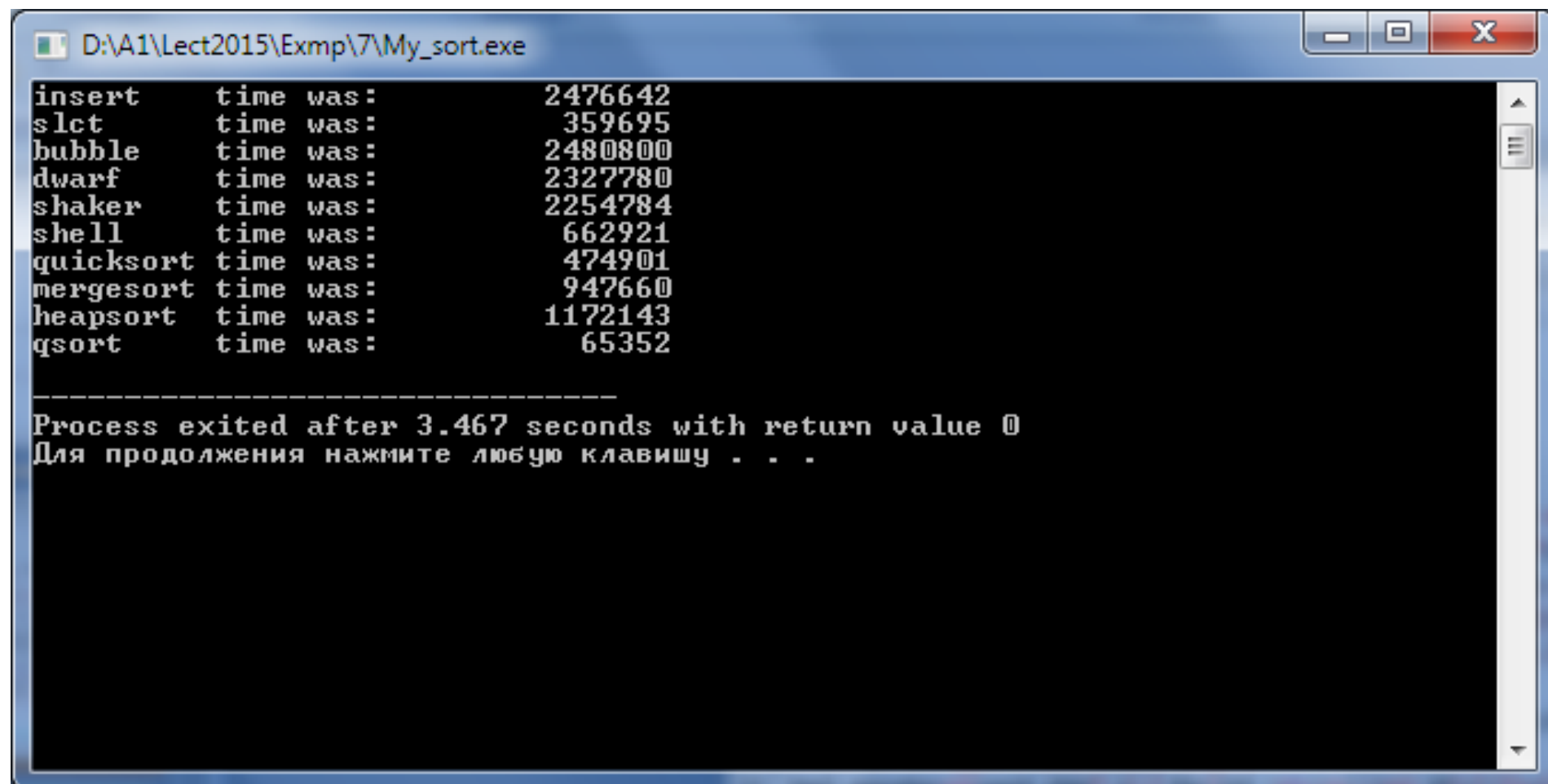
Компилятор Ресурсы Журнал компиляции Отладка Результаты поиска

Line: 13 Col: 1 Sel: 0 Lines: 52 Length: 1401 Вставка Done parsing in 0,062 seconds

```

1 #include <string.h>
2 #include <malloc.h>
3 class My_string { char* s; int l;
4 public:
5 My_string(const char* p1=NULL, const char* p2=NULL) { char* b;
6     if(p1 == NULL ) { s = NULL; l = -1; return; } else if(p2 == NULL) { p2 = p1; while(*p2++); p2--; }
7     l = p2 - p1; b = s = new char[l+1]; while(p1<p2) *b++ = *p1++; *b='\0'; }
8 My_string(const My_string& x) { s = new char[l = x.l]; strcpy(s,x.s); }
9 void operator=(const char* x) { delete s; s = new char[(l=strlen(x))+1]; strcpy(s,x); }
10 void operator=(const My_string& x) { delete s; s = new char[(l=x.l)+1]; strcpy(s,x.s); }
11 My_string operator+(const My_string& a) {My_string loc;
12     loc.s = new char [(loc.l+l+a.l)+1]; strcpy(loc.s,s); strcat(loc.s,a.s); return loc; }
13 char& operator[](int i) { return s[i]; }
14 bool operator==(const char* x) { for(int i=0;i<=l;i++) if(s[i]!=x[i]) return false; return true; }
15 bool operator<(const My_string& x) { for(int i=0;i<=l;i++) if(s[i]==x.s[i]) continue; else return s[i]<x.s[i]; return false; }
16 bool operator>(const My_string& x) { for(int i=0;i<=l;i++) if(s[i]==x.s[i]) continue; else return s[i]>x.s[i]; return false; }
17 int size() { return l; }
18 int find(const My_string& x) { for(int i=0; i<l; i++) if(!strncmp(s+i,x.s,x.l)) return i; return -1; }
19 friend ostream& operator<<(ostream& a, const My_string& x) { return a << x.s << "\n"; }
20 friend istream& operator>> (istream& b, My_string& a) { int l=0; const int n=20;char s[n]; s[n-1]='\0'; delete a.s;
21     *(a.s = new char)='\0'; a.l=0; if(b.get(s[l])) b.putback(s[l]); else return b;
22     while(b.get(s[l]),s[l++]!='\n') if(l==n-1) { l=0; a.s=(char*)realloc(a.s,a.l+=n); strcat(a.s,s); }
23     if(l>1) { s[l-1]='\0'; a.s=(char*)realloc(a.s,a.l+=l); strcat(a.s,s); }
24     return b; }
25 ~My_string() { delete s; l = -1; }
26 };
27

```



```
D:\A1\Lect2015\Exmp\7\My_sort.exe

insert    time was:      2476642
slct      time was:      359695
bubble    time was:      2480800
dwarf     time was:      2327780
shaker    time was:      2254784
shell     time was:      662921
quicksort time was:      474901
mergesort time was:      947660
heapsort  time was:     1172143
qsort     time was:      65352

-----
Process exited after 3.467 seconds with return value 0
Для продолжения нажмите любую клавишу . . .
```

Библиотечная функция qsort

```
void qsort(void* a, size_t n, size_t s,  
int (*comp)(const void*, const void*))
```

здесь

a – имя сортируемого массива

n – количество элементов a

s – размерность одного элемента

(*comp) – указатель на функцию сравнения

> 0, если >

= 0, если ==

< 0, если <

```
int comp(const void* i,const void* j)
{
    double r = *(double*)i-*(double*)j;
    return r>0?1:(r<0?-1:0);
}
```

```
int main() {
    init(a);
    qsort(a,N,sizeof(double),comp);
    return 0;
}
```