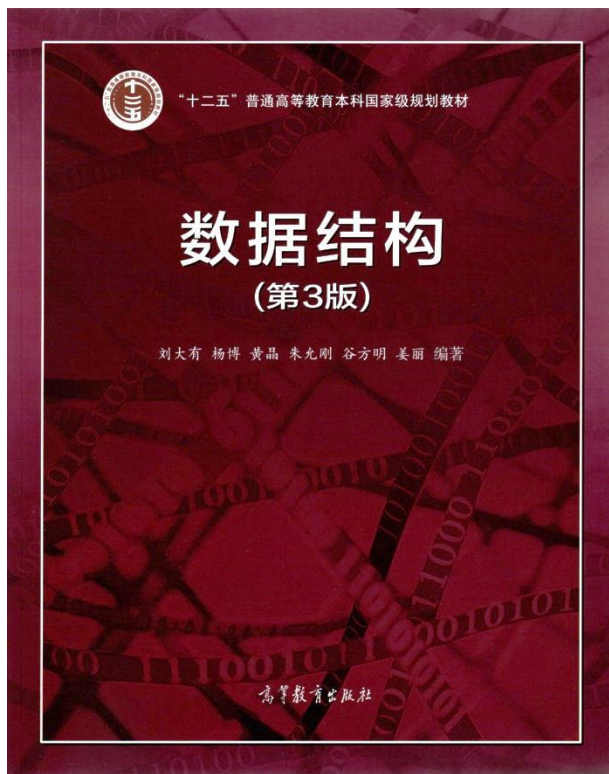




计算机学院王湘浩班
2024级



堆与优先级队列

- 堆的插入删除
- 优先级队列及应用
- Top K问题

数据之法
结构之美
算法之道

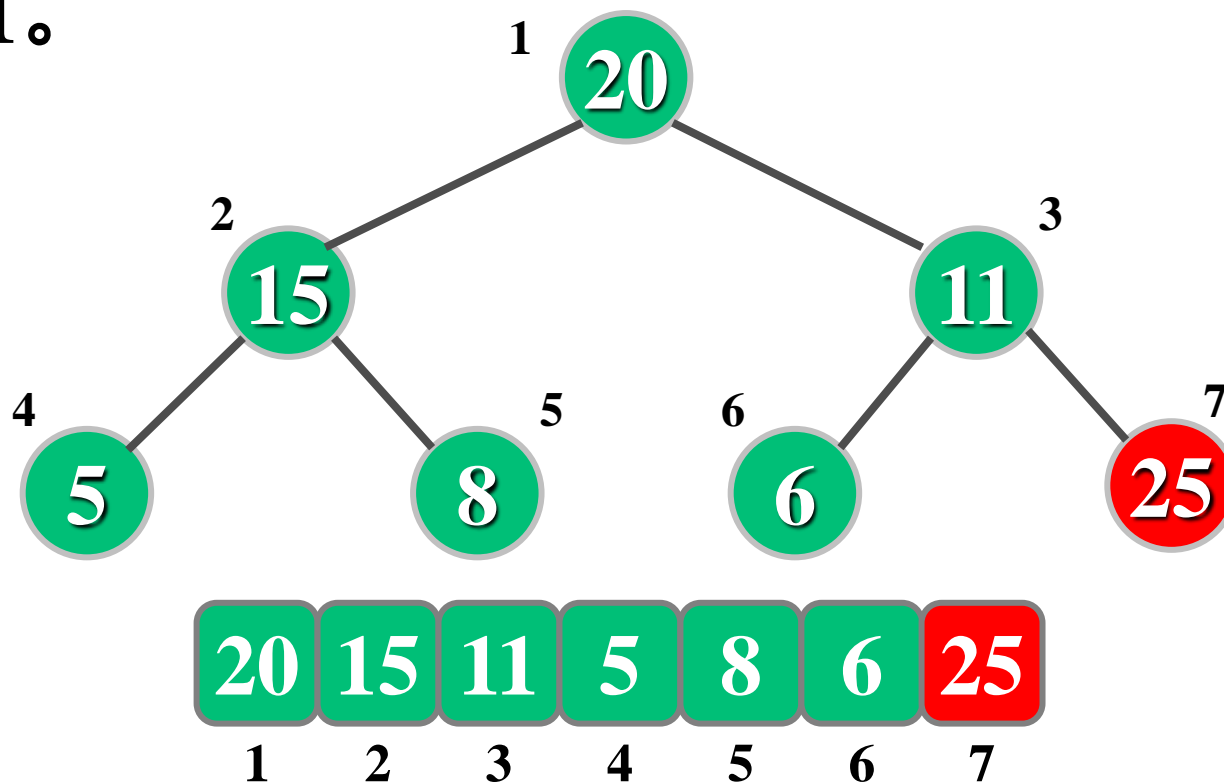
Last updated on 2025.6

zhuyungang@jlu.edu.cn



堆的插入操作

- 目的：(R_1, \dots, R_n)是一个堆，插入一个新元素 R_{n+1} ，使(R_1, \dots, R_{n+1})成为一个堆。
- 做法：把新插入的元素放在堆尾，再对其做上浮操作，堆的元素个数加1。



插入元素
25

堆的插入操作

```
void Insert(int R[], int &n, int x){ //堆尾插入值x  
    R[++n] = x;           //x放在R[n+1]处，堆元素个数加1  
    ShiftUp(R, n, n); //元素R[n]上浮  
}
```

时间复杂度
 $O(\log n)$

课下思考

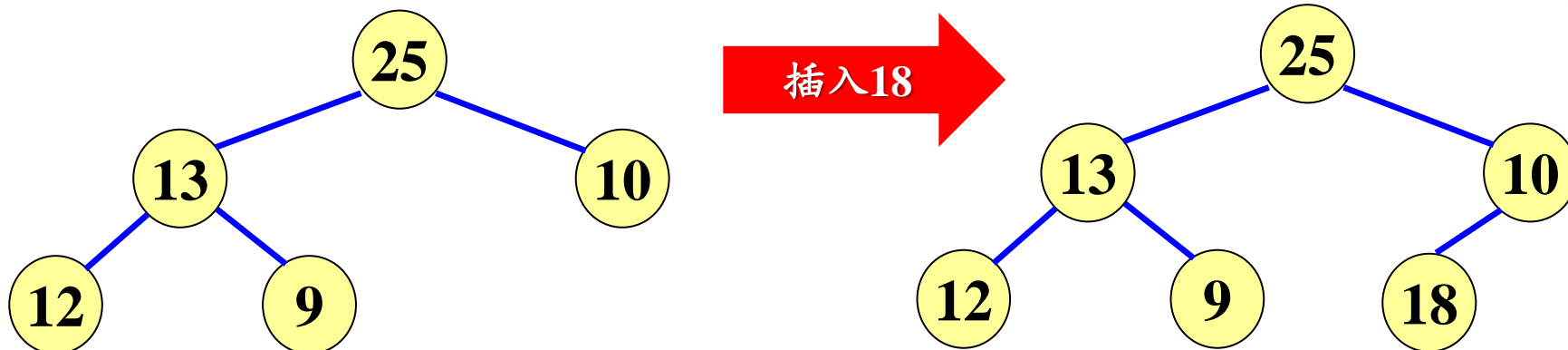
已知序列(25, 13, 10, 12, 9)是大根堆，在此序列尾部插入新元素18，将其再调整为大根堆，调整过程中元素比较次数为_____。【考研题全国卷】

[A] 1

[B] 2

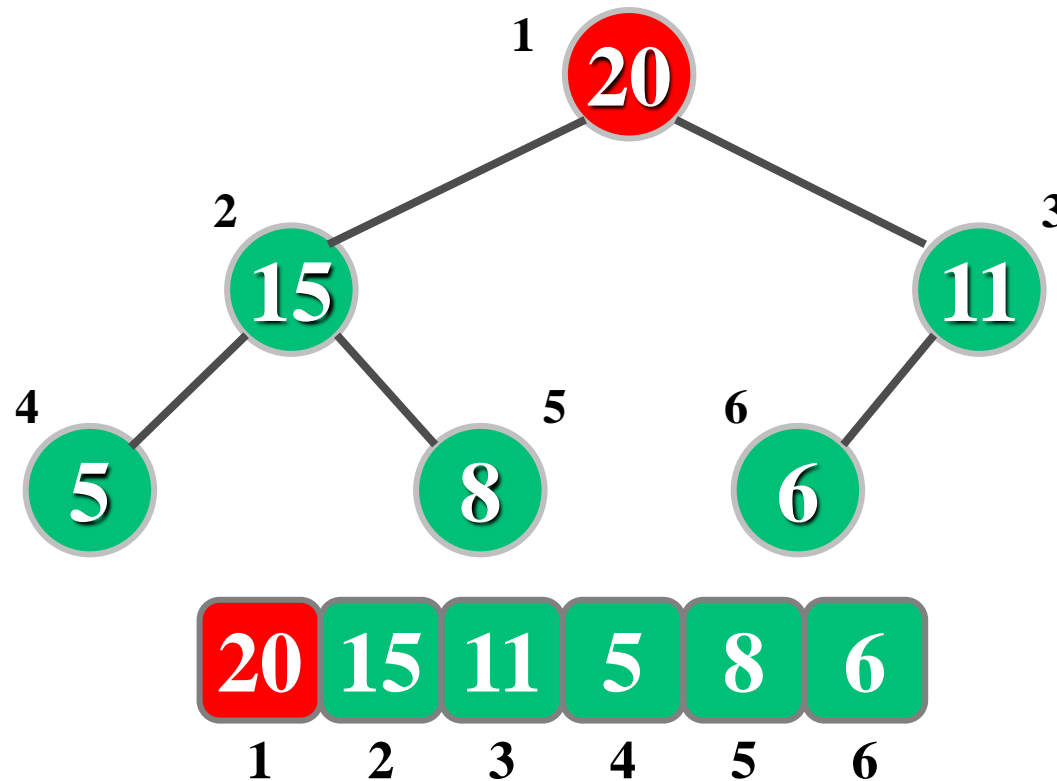
[C] 4

[D] 5



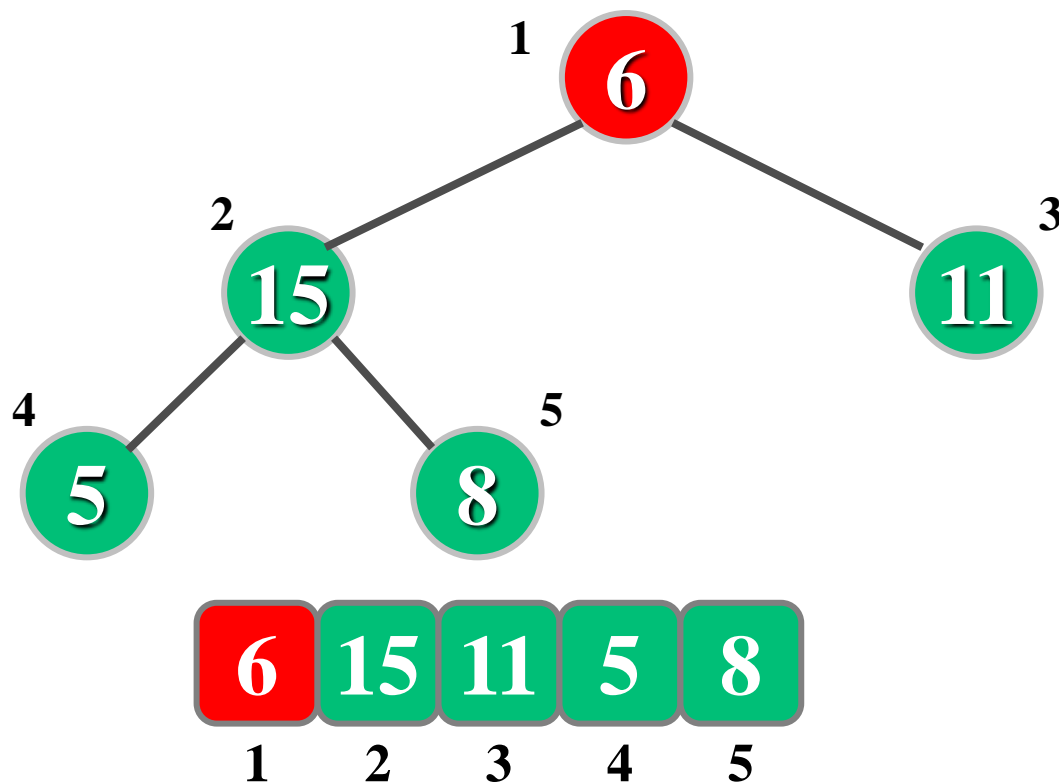
堆的删除操作

- 目的: (R_1, R_2, \dots, R_n) 是一个堆, 从堆中删除并返回堆顶 (即取出堆内最大的元素), 且删除后的文件仍为堆。
- 做法: 把 $R[n]$ 移到根, 堆元素个数减1, 再对根 $R[1]$ 做 **下沉** 操作。



堆的删除操作

- 目的: (R_1, R_2, \dots, R_n) 是一个堆, 从堆中删除并返回堆顶 (即取出堆内最大的元素), 且删除后的文件仍为堆。
- 做法: 把 $R[n]$ 移到根, 堆元素个数减1, 再对根 $R[1]$ 做 **下沉** 操作。



堆的删除操作

```
int DelMax(int R[], int &n){ //删除并返回堆顶, 假设堆非空
    int MaxKey=R[1]; //暂存堆顶元素
    R[1] = R[n--]; //堆尾移至堆顶, 堆元素个数减1
    ShiftDown(R, n, 1); //新堆顶R[1]下沉
    return MaxKey;
}
```

时间复杂度
 $O(\log n)$

课下思考

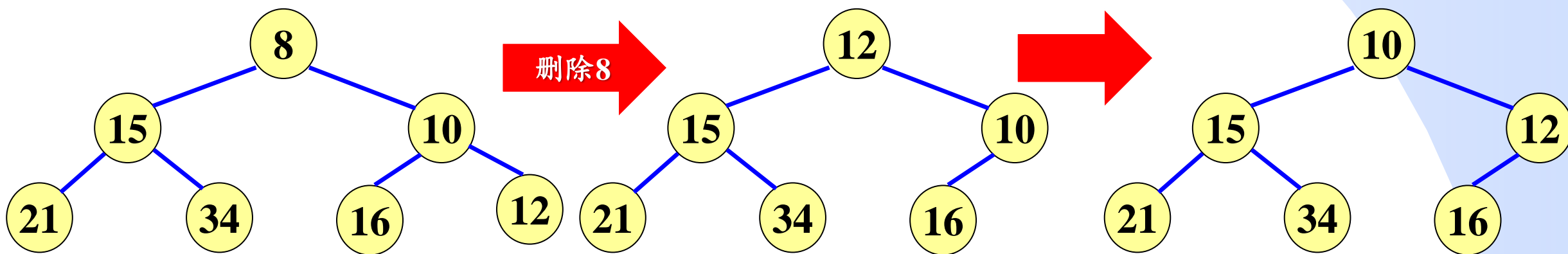
已知小根堆(8, 15, 10, 21, 34, 16, 12), 删除关键字8后需要重建堆, 在此过程中, 关键词比较次数为_____。【考研题全国卷】

[A] 1

[B] 2

[C] 3

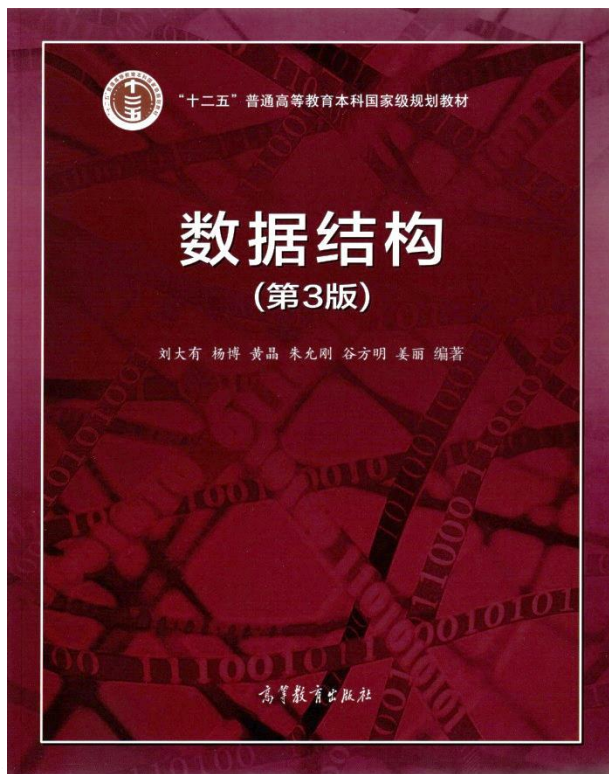
[D] 4





堆与优先级队列

- 堆的插入删除
- **优先级队列及应用**
- Top K问题



数据之法
结构之美
算法之道

优先级队列 (Priority Queue)

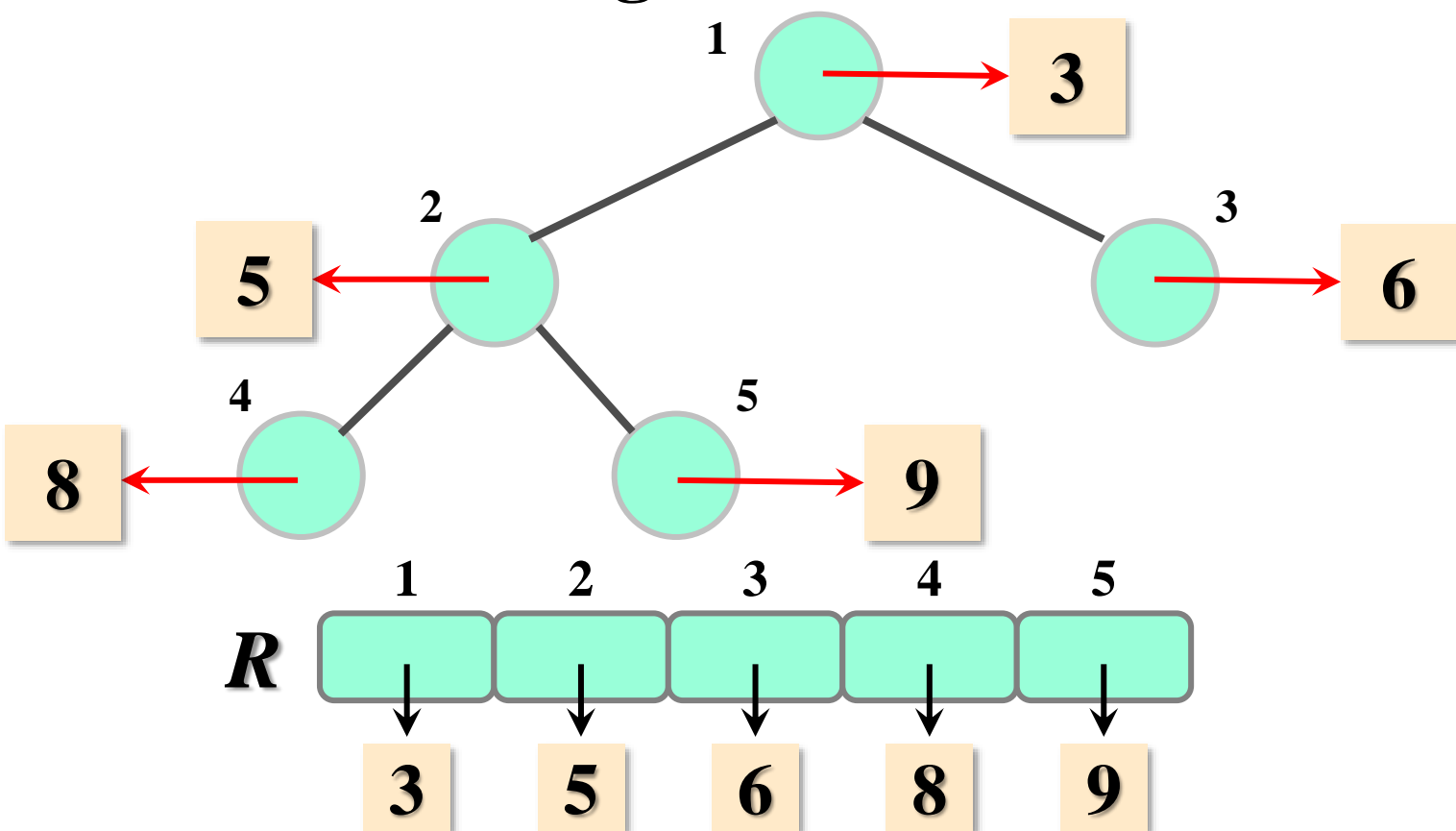
- 为每个元素设置一个优先级。
- 出队顺序：优先级高的元素先出队，优先级低的元素后出队。
- 可以用堆实现优先级队列，优先级对应关键词大小。
- 入队：堆的插入，时间 $O(\log n)$
- 出队：堆的删除，时间 $O(\log n)$
- 可以把优先级队列实现成类，插入、删除等操作作为成员函数，存储堆的数组 R 作为数据成员。

```
const int maxn=1e5+10;
Template<class T>
class PriorityQueue{
public:
    void Insert(T K);
    T DelMax();
    bool Empty();
    .....
private:
    void ShiftUp(int i);
    void ShiftDown(int i);
    T R[maxn]; //存堆元素
    int n; //堆的元素个数
    .....
};
```

优先级队列（最小堆）优化Huffman算法

- 堆的元素：哈夫曼树的结点指针（即 `HuffmanNode*` 类型）
- 堆元素 $R[i]$ 的关键词： $R[i] \rightarrow \text{weight}$
- 用堆选取 `weight` 最小的结点

```
const int maxn=1e5+10;
Template<class T>
class MinPQ{
public:
    void Insert(T K);
    T DelMin();
    bool Empty();
    ....
private:
    void ShiftUp(int i);
    void ShiftDown(int i);
    T R[maxn]; //存堆元素
    int n; //堆的元素个数
    ....
};
```



优先级队列（最小堆）优化Huffman算法

```
MinPQ<HuffmanNode*> pq;           //基于最小堆创建优先级队列
```

```
... ..
```

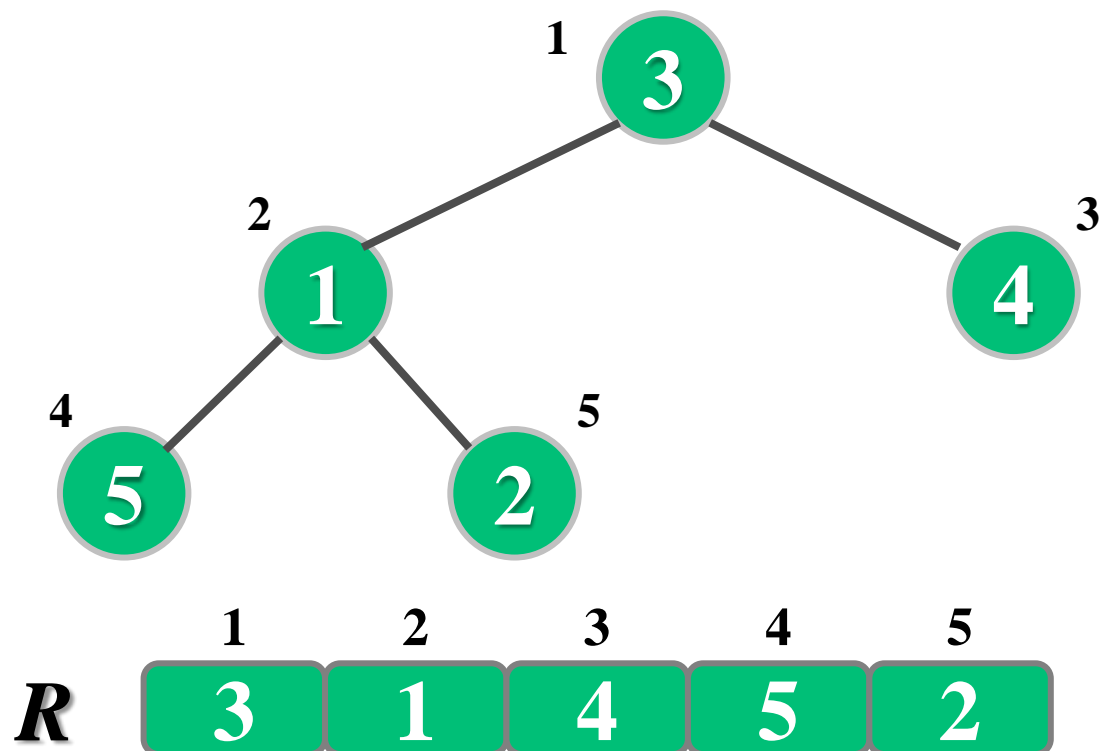
```
for (int i=0; i<n-1; i++){  
    HuffmanNode *t = new HuffmanNode;  
    t->left = pq.De1Min(); //取出权值最小的结点  
    t->right= pq.De1Min(); //取出权值最小的结点  
    t->weight = t->left->weight + t->right->weight;  
    pq.Insert(t);          //新创建的结点插入堆  
}
```

时间复杂度
 $O(n\log n)$

用堆选取weight
最小的结点

优先级队列（最小堆） 优化Dijkstra算法

- 堆的元素：图中顶点的编号
- 堆元素 v 的关键词： $\text{dist}[v]$
- 用堆选取 dist 值最小的顶点



```
const int maxn=1e5+10;
Template<class T>
class MinPQ{
public:
    void Insert(int v, T key);
    int DelMin();
    bool Empty();
    .....
private:
    void ShiftUp(int i);
    void ShiftDown(int i);
    int R[maxn]; //存堆元素
    T dist[maxn]; //存关键词
    int n; //堆的元素个数
    .....
};
```


堆优化的Dijkstra算法

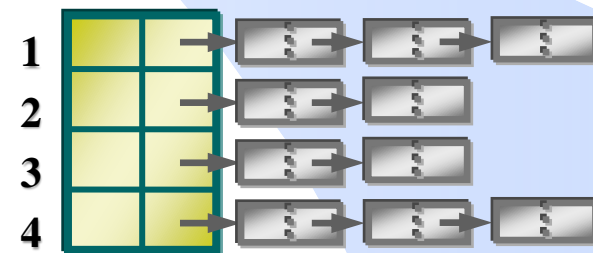
```

void Dijkstra(Vertex *Head, int n, int u, int dist[]){
    int S[N]={0}; MinPQ<int> pq;
    for(int i=1; i<=n; i++) dist[i]=(i==u)?0:INF;
    pq.Insert(u, 0);
    while(!pq.Empty()) {
        int v = pq.DeMin(); if(S[v]==1) continue;
        S[v]=1; //将顶点v放入S集合
        for(Edge* p=Head[v].adjacent; p; p=p->link){
            int w=p->VerAdj; //更新v的各邻接顶点w的dist值
            if(S[w]==0 && dist[v]+p->cost < dist[w]){
                dist[w] = dist[v]+p->cost;
                pq.Insert(w, dist[w]);
            }
        }
    }
}

```

时间: $e \cdot T(\text{DeMin}) + e \cdot T(\text{Insert})$

课下思考：时间复杂度 $O(e \log e)$ 。提示：堆内最多可能有 e 个顶点。适合稀疏图。



(5, 13)

(5, 20)

Dijkstra
基于优先级队列的BFS

课下阅读：C++ STL实现堆优化的Dijkstra算法

- **priority_queue**: C++ STL内置的优先级队列，默认为最大堆。若需要最小堆，可对关键词取负存入堆中。常用方法如下：

方法	功能	时间复杂度
push()	把元素插入堆	$O(\log n)$
pop()	删除堆顶元素	$O(\log n)$
top()	返回堆顶元素	$O(1)$
size()	返回堆元素个数	$O(1)$

- **pair<>**: C++ STL内置的二元组，尖括号中分别指定二元组的第一元、第二元的类型，可以用**make_pair**函数创建二元组，用成员变量**first**访问第一元、**second**访问第二元。在比较大小时，以第一元为第一关键词、第二元为第二关键词。

课下阅读：C++ STL实现堆优化的Dijkstra算法

```

void Dijkstra(Vertex *Head, int n, int u, int dist[]){
    int S[N]={0};    priority_queue<pair<int,int>> pq;
    for(int i=1; i<=n; i++) dist[i]=(i==u)?0:INF;
    pq.push(make_pair(0, u));
    while(pq.size()) {
        int v = pq.top().second(); pq.pop();
        if(S[v]==1) continue;
        S[v]=1;    //将顶点v放入S集合
        for(Edge* p=Head[v].adjacent; p!=NULL; p=p->link){
            int w=p->VerAdj; //更新v的各邻接顶点w的dist值
            if(S[w]==0 && dist[v]+p->cost < dist[w]){
                dist[w] = dist[v]+p->cost;
                pq.push(make_pair(-dist[w], w));
            }
        }
    }
}

```


斐波那契堆优化的Dijkstra算法

	<i>DelMin</i>	<i>Insert</i>	<i>DecreaseKey</i>	<i>Dijkstra</i> 算法
斐波那契堆	$O(\log n)$	均摊 $O(1)$	均摊 $O(1)$	$O(n \log n + e)$



Robert Tarjan
图灵奖获得者
普林斯顿大学教授
美国科学院/工程院院士
提出斐波那契堆

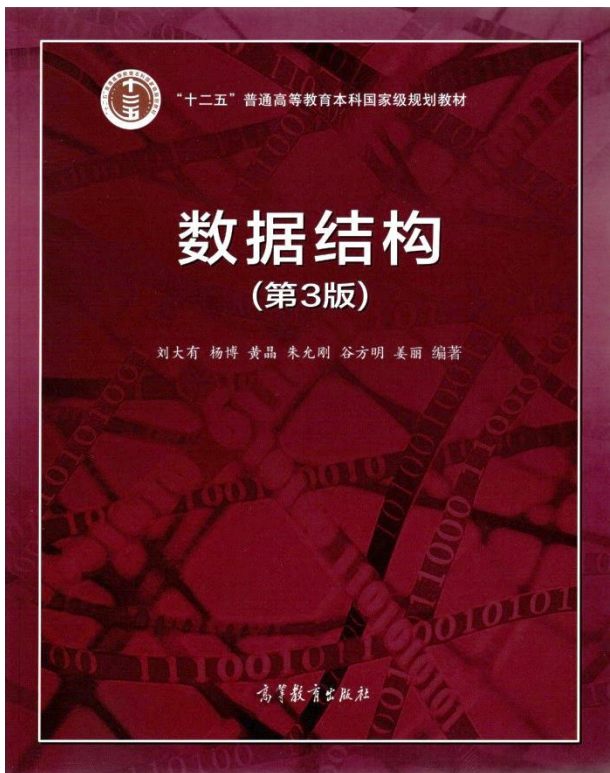
图的搜索策略

- **DFS**: *Depth First Search* (深度优先搜索)
- **BFS**: *Breath First Search* (广度优先搜索)
- **PFS**: *Priority First Search* (优先级优先搜索)
 - ✓ Dijkstra算法: 优先级—*dist*
 - ✓ Prim算法: 优先级—*Lowcost*

图搜索的一种统一框架

```
void GraphSearch(Graph G, int u) { //在图G中以u为起点进行某种搜索
    int visited[N]={0};
    CONTAINER container; //定义一个容器
    container  $\leftarrow$  u;      //把起点放入容器
    while(!container.Empty()) {
        int v  $\leftarrow$  container; //从容器中取出点v
        if(visited[v]==1) continue;
        visit(v); visited[v]=1;
        for(each w in neighbours(v)) //考察v的每个邻接顶点w
            if(visited[w]==0 && ...) //若w未被处理过且满足某些条件
                container  $\leftarrow$  w;
    }
}
```

容器	搜索策略
栈	<i>DFS</i>
队列	<i>BFS</i>
优先级队列	<i>PFS</i>



堆与优先级队列

- 堆的插入删除
- 优先级队列及应用
- **Top K问题**

数据之法
结构之美
算法之道

热搜榜

Baidu 百度 数据结构 数据结构 重要性 百度一下

邓满英 - 《辽宁行政学院学报》 - 2008
《数据结构》是一门基础学科,能让大家进一步理解各种数据对象的特点,学会数据的组织方法和实现方法。这门课程着眼是培养大家的抽象思维、创造能力和逻辑思维能力,使大...
xueshu.baidu.com

[数据结构的重要性 - lczi - 博客园](#)
2014年2月24日 - 数据结构的重要性 数据结构(计算机存储、组织数据方式) 数据结构是计算机存储、组织数据的方式。数据结构是指相互之间存在一种或多种特定关系的数据...
https://www.cnblogs.com/jsjblc... - 百度快照

[学习数据结构的意义和作用_百度文库](#)
★★★★★ 评分:5/5 6页
2012年3月24日 - 学习数据结构的意义和作用董建寅 罗远(上海金融学院...应用专业,电子商务等专业的 基础课,是十分重要的...据统计,当今处理非数值计算性问 题占用了 85...
百度文库 - 百度快照

[数据结构对于编程语言的重要性,你了解多少? 算法](#)
2018年11月21日 - 数据结构是一门综合性较强的计算机软件、程序设计理论和技术相结合的重要基础知识。它主要讨论抽象数据关系和算法在计算机中的表示与实现,涉及到的数据在计算机中的...
搜狐网 - 百度快照

搜索热点 [换一换](#)

1 临港新片区揭牌	852万
2 张予曦承认分手	757万 ↑
3 地球快没沙子了	715万 ↑
4 小海绵扮消防员 新	643万
5 新LPR形成机制 新	603万 ↑
6 曼联战平狼队	570万
7 深圳被委以重任	542万 ↑
8 运营商否认4G降速 新	523万
9 具惠善安宰贤离婚	507万 ↑
10 哪吒票房破41亿	455万 ↑

[查看更多>>](#)

数组有 n 个元素，挑选其中最大的 k ($k < n$)个元素。

TOP K问题

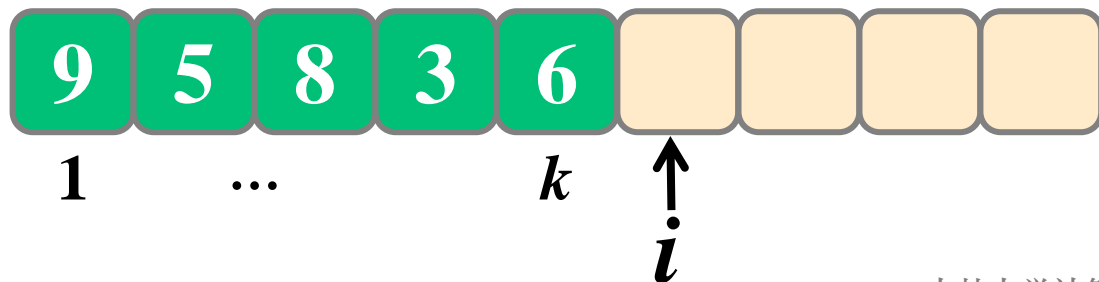
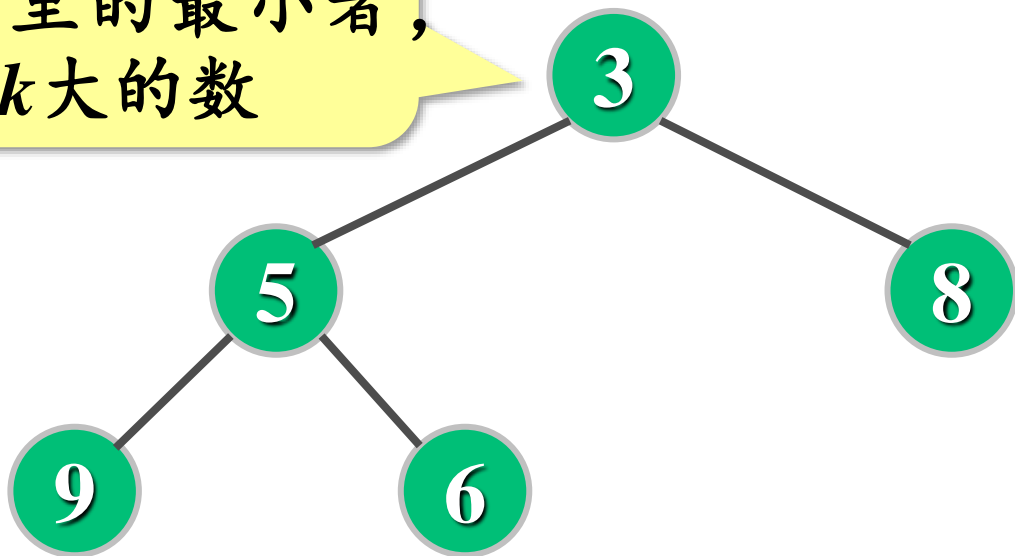
数组有 n 个元素，挑选其中最大的 k ($k < n$)个元素。【华为、腾讯、字节跳动、阿里、京东、美团、苹果、微软、谷歌面试题】

- 想法1：递减排序后，选前 k 个数， $O(n\log n)$ 。
- 想法2：借助直接选择排序思想，选 k 次最大元素， $O(nk)$ 。
- 想法3：借助堆，取出 k 次堆顶元素， $O(n+k\log n)$ 。

TOP K问题

使用包含 k 个元素的小根堆维护数组中最大的 k 个元素。

堆顶是目前Top k 元素里的最小者，即第 k 大的数



- ① 用数组前 k 个元素建堆，此时堆里存的就是当前扫描过的元素里的最大的 k 个元素。
- ② 依次扫描数组剩余元素，若扫描到的元素 x 大于堆顶，则用 x 替换堆顶，下沉 x 以重建堆。使堆始终保存当前扫描过的元素里的Top k 元素。

时间复杂度 $O(n \log k)$
适合增量环境