TG4 初级数据结构 1

anantheparty cdqz->hust

HINT

- 课程内容涉及提高中用的到的数据结构和一些和数据结构沾了边的算法
- 题目难度由易到难，受众群体有一定差异，自行斟酌完成题目
- 课后完成上课所讲的能力范围内的例题（建议）

初级数据结构 1

- 并查集
- 倍增
  - RMQ（ST表）
- 单调栈&单调队列
- 堆
  - 优先队列
  - 堆排序
- Trie树

一些概念

- 离线/在线算法（提高组不常见）：
  - 在线：每次询问后，立马可以得到查询结果
  - 离线：需要知道所有要查询的值，然后一口气查询出所有结果

一些基础知识

- STL中set的用法

```
#include<set>
set<int>S;
S.insert(x);S.erase(x);S.size();S.empty();
set<int>::iterator i;
i=S.upper_bound(x);i=S.lower_bound(x);
i=S.find(x);if(i!=S.end())…;
for(i=S.begin();i!=S.end();i++)…;
*i
```

- S.lower_bound(x) 表示查找 >= x 的元素中最小的一个，并返回指向该元素的迭代器
- S.upper_bound(x) 表示查找 >x 的元素中最小的一个，并返回指向该元素的迭代器

并查集

- 支持一些不相交集合的合并和查询
- 初始时，每个元素 x 各有一个集合 {x}
- 两种操作：
  - 查找：查询某元素 x 属于哪一个集合（判断两个元素 x 和 y 是否处于同一集合）
  - 合并：将元素 x 所在的集合和元素 y 所在的集合合并

并查集

- 使用树形结构组织数据：
  - 同一集合内的元素构成一棵树
  - 因为集合间没有交，所以最终构建得到的是一个森林
- 将树的根节点作为集合的代表元
- 初始时：n 个孤立的点

并查集

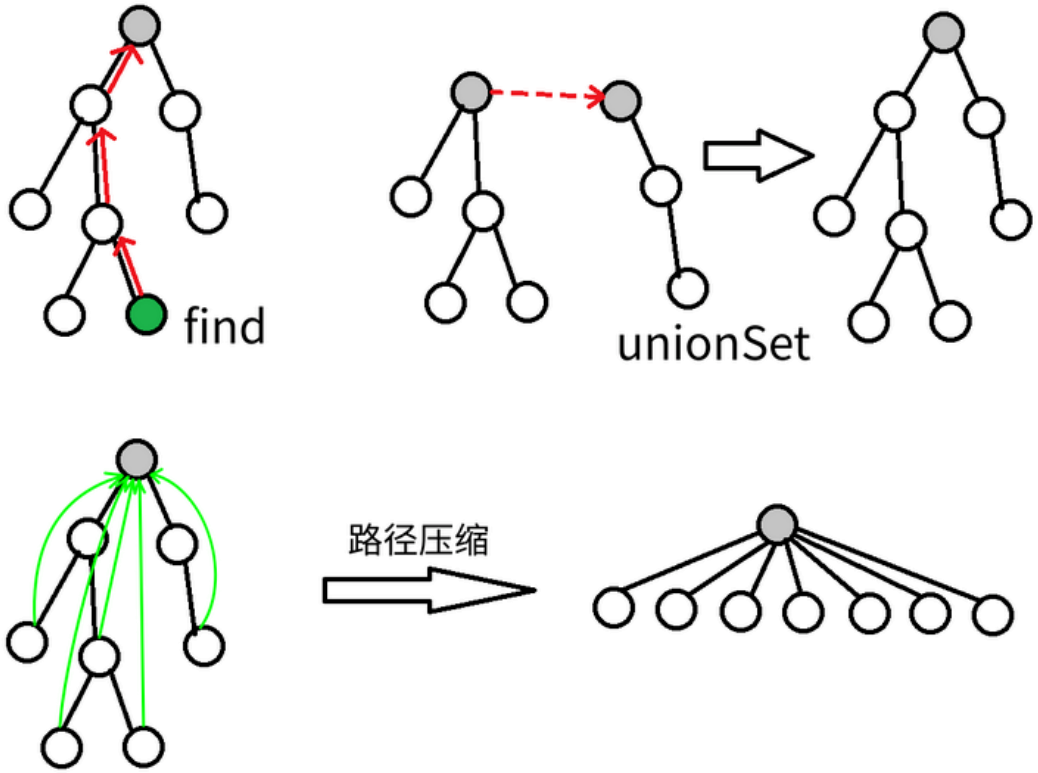
- 查询（Find）操作
  - 寻找 x 的根节点，即集合的代表元素

```
int find(int x) {
    return x == fa[x] ? x : fa[x] = find(fa[x]);
}
```

- 使用路径压缩，加快查询速度

并查集

- 路径压缩
- 把在路径上的每个节点都直接连接到根上，这就是路径压缩。



并查集

- 合并（Union）操作
  - 寻找 x 和 y 的根节点，将其中一个挂到另一个下面

```
void merge(int x, int y) {
    x = find(x);
    y = find(y);
    if (x != y) fa[x] = y;
}
```

最小生成树

- Kruskal 算法：为了造出一棵最小生成树，我们从最小边权的边开始，按边权从小到大依次加入，如果某次加边产生了环，就扔掉这条边，直到加入了n-1条边，即形成了一棵树。
- 使用并查集判断是否成环

P3958

- Link

P1197

- Link

常见套路：P1396

- Link
- 类似题目： P1783等

单调栈&单调队列

- 单调栈即满足单调性的栈结构，其只在一端进行进出
- 单调**队列**即满足单调性的**队列**结构，通常队首只能出，队尾可以进出（deque）

单调栈&单调队列

- 如何维护
- 一句话：push时将所有不满足单调性的点pop

单调栈&单调队列

- 单调栈的本质是单调队列的退化(栈底不能出栈)
- 单调栈可以求出，每个数(前/后)方比它(小/大)的第一个数，以及任意前缀最大(最小)值
- 单调队列可以求出，一个滑动区间内，每个数(前/后)方比它(小/大)的第一个数，以及该区间内最大(最小)值
- 可以看出，求单调栈任意前缀最大(最小)值可以前缀和直接处理
- 求每个数(前/后)方比它(小/大)的第一个数时往往没有区间约束
- 因此单调栈和单调队列解决的问题往往不同

单调栈&单调队列

- 给定 n 个非负整数表示每个宽度为 1 的柱子的高度图，计算按此排列的柱子，下雨之后能接多少雨水

P1106

- Link

P1823

- Link

P1823

- Link

P3957

- Link

倍增

- 简单来说倍增是一种每次翻倍的计算方式，任何翻倍算法都可以算倍增，常见的是翻两倍的二倍增
- 倍增常见的应用是通过当前所有节点的2^i的状态，推算出2^(i+1)的状态
- 提高组常见的倍增求LCA会计算每个节点向上第2^i个父亲，同时推断出第2^(i+1)个父亲

倍增求RMQ

- RMQ 是英文 Range Maximum/Minimum Query 的缩写，表示区间最大（最小）值
- 可以利用倍增算法，构造ST表，O(nlogn)预处理，O(1)查询，空间复杂度O(nlogn)

ST表的构造与使用

- 我们有一个长度为n的序列，我们设f[i][x]表示第x个数开始，长度为2^i的一段数的最值
- 可以得到f[i][x]=min/max(f[i-1][x],f[i-1][x+2^(i-1)])
- 利用f[i-1]推导f[i]
- 对于查询[l,r]的最值，我们只需要选择两个区间的合为[l,r]即可，即若2^i<=r-l+1<=2^(i+1)，我们选取两个长度为2^i的区间，一个左侧对齐l，一个右侧对齐r，求这两个区间的最值即可
- 模板题P3865

P1081 开车旅行

- Link

P7167

- Link

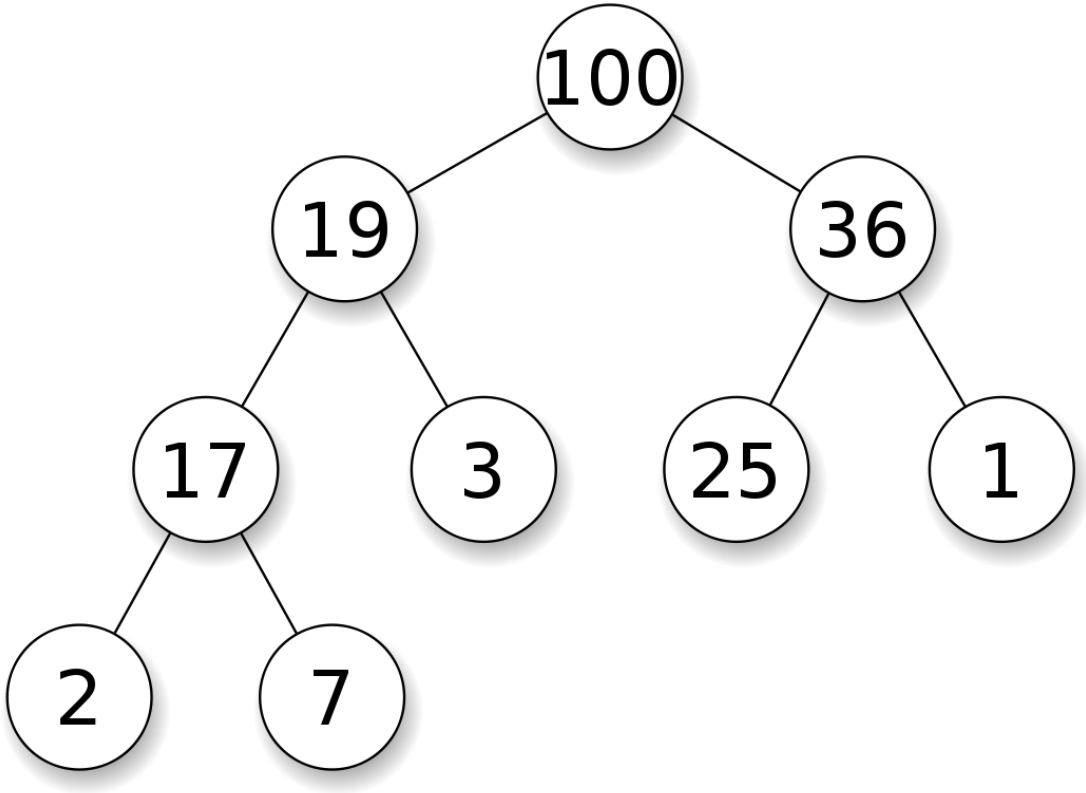
P1081 开车旅行

- 预处理A和B在每一个城市的决策：使用set直接计算，使用双向链表巧妙计算
- 使用dp处理所有情况
- f[k][i][j]表示以i为起点，k第一个开车，开车j天，到达的目的地
- fa[k][i][j] fb[k][i][j]表示以i为起点，k第一个开车，开车j天，A和B分别行使距离
- 使用倍增优化，j表示开车2^j天

堆

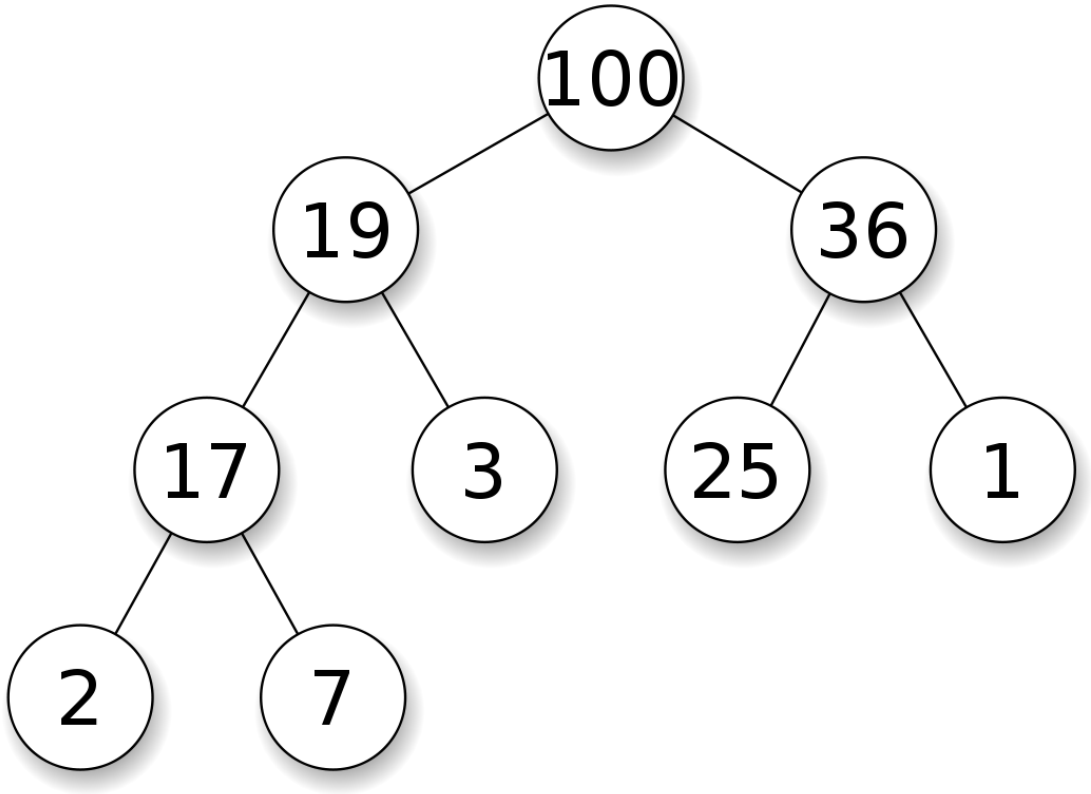
- 大根堆：父亲一定比儿子大的完全二叉树
- 小根堆：父亲一定比儿子小的完全二叉树

- 根节点永远是最大（最小）的
- 并不符合二叉搜索树



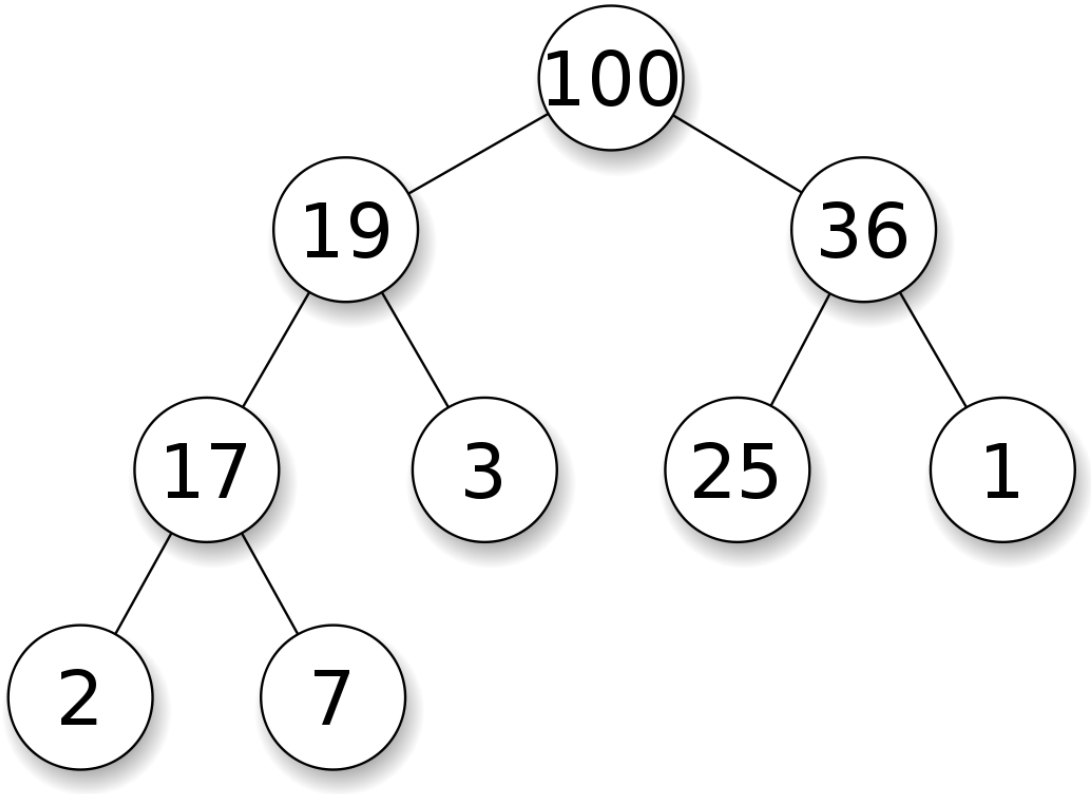
堆

- 插入：在当前堆的最后插入，向上调整到根



堆

- 删除：只能删除跟节点
- 将最后节点移动到根后向下调整

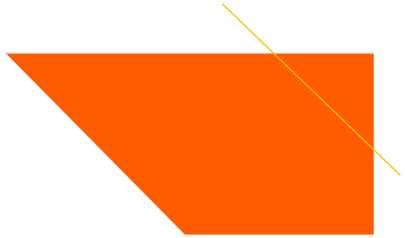


优先队列

- 优先队列的本质就是堆

```
#include<queue>
std::priority_queue<int> Q;
Q.top();Q.pop();Q.push(x);Q.size();Q.empty();
```

- 这样每次top查询的是队列中最大的数
- push和pop是O(logn)的，top为O(1)



优先队列

- 手写?
- 用堆实现
- 支持删除和插入两个操作即可

优先队列

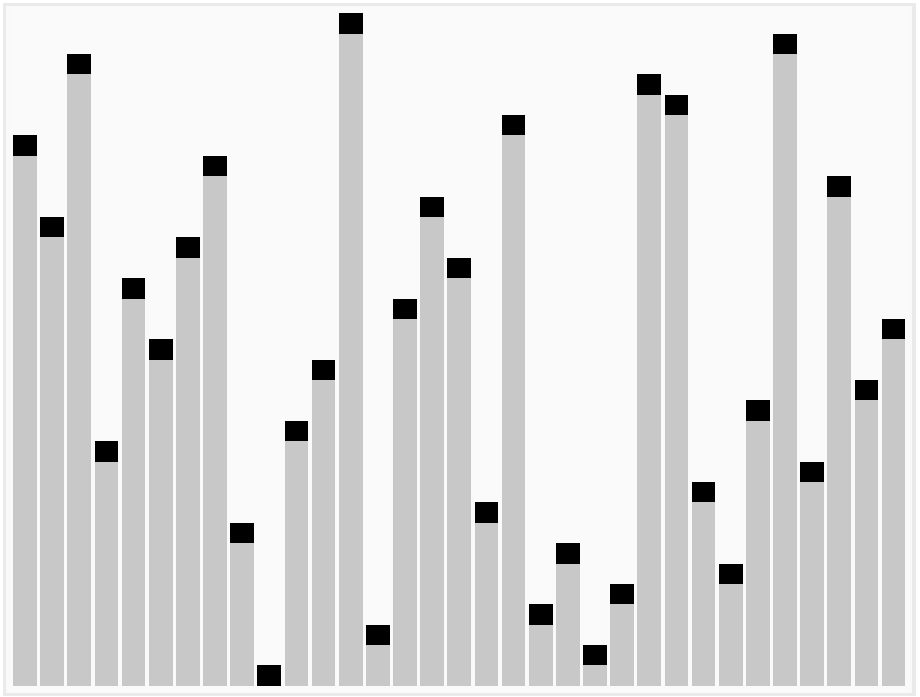
- 如何使每次top查询的是最小的数?
- priority_queue<Type, Container, Functional>

- priority_queue<int,vector<int>,greater<int> >Q;
- priority_queue<pair<int,int> >,vector< pair<int,int> >,greater< pair<int,int> > >Q;

  (Dijkstra堆优化通常使用)

堆排序

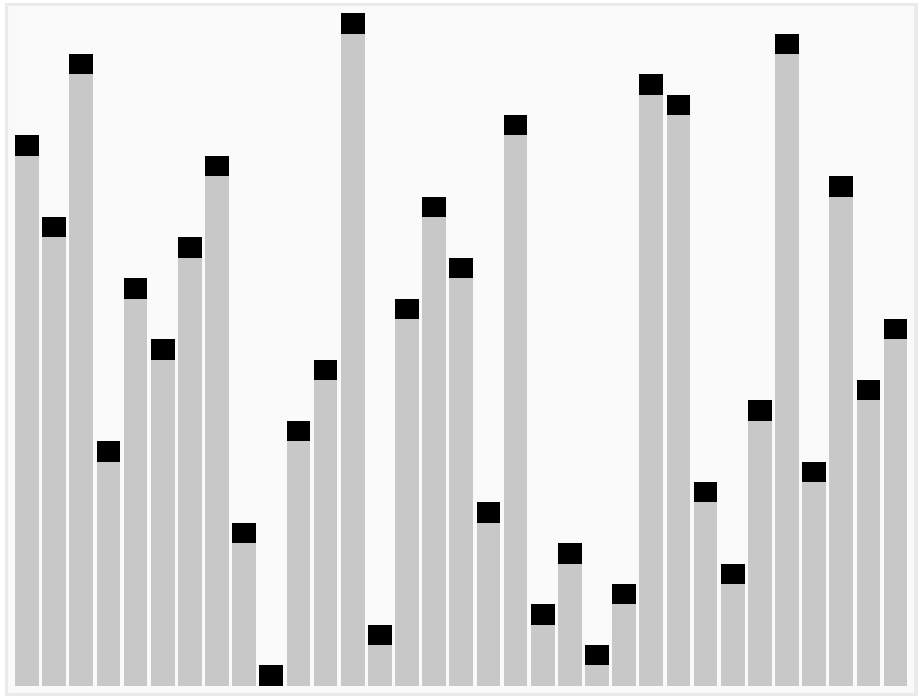
- 利用堆的根节点最大和完全二叉树性质的排序

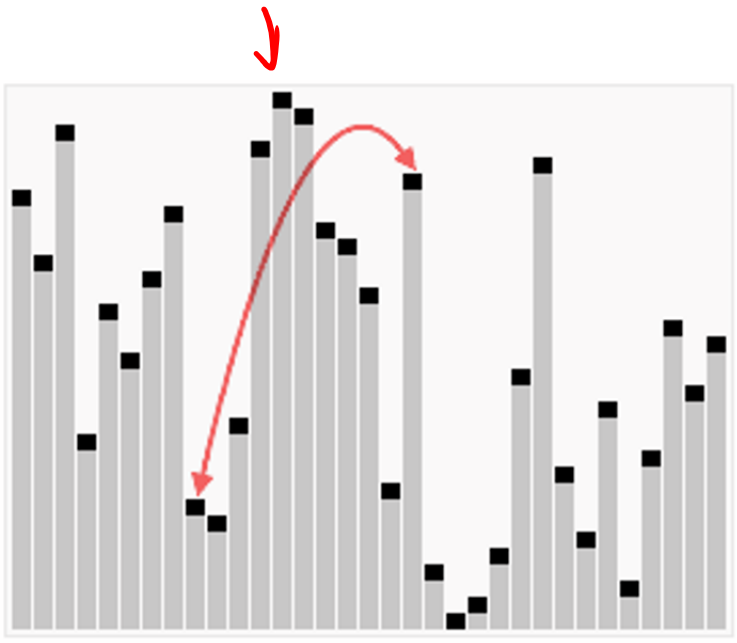


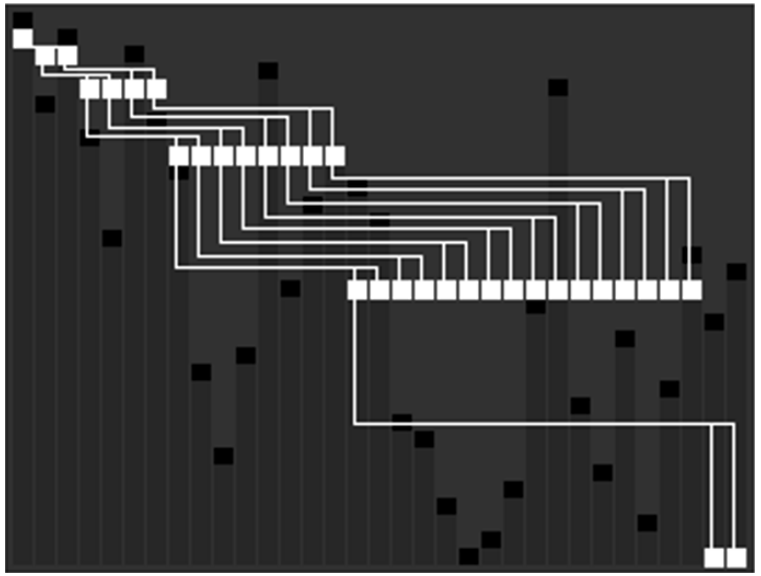
-------From Wikipedia

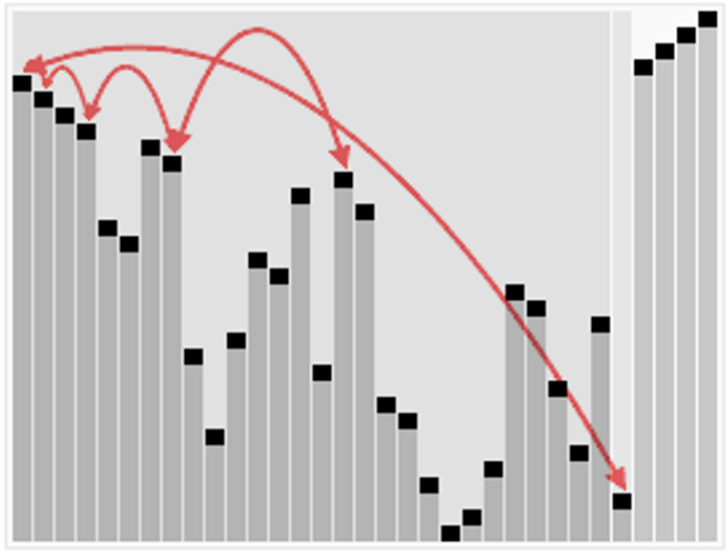
堆排序

- 利用堆的根节点最大和完全二叉树性质的排序
- 主要有两个部分：
- 建立一个大根堆
- 不停的取出堆顶元素







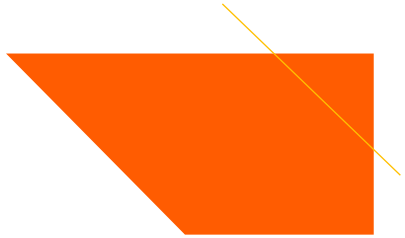


堆排序

```
void update(int x,int m)
{
    int t=lson(x);
    if(t>m)return ;
    if(t<m&&a[t+1]>a[t])t++;
    if(a[t]>a[x])swap(a[x],a[t]);
    update(t,m);
}

void heap_sort()
{
    for(int i=n;i>=1;i--)update(i,n);//建立大根堆
    for(int i=n-1;i>=1;i--)
    {
        swap(a[i+1],a[1]);  //依次取出
        update(1,i);
    }
}
```

- 下发文件附完整代码



堆排序

- 应用?
- 找topK
- 复杂度O(n+k*log(n))

2016D2T2 P2827 蚯蚓

Link

2016D2T2 P2827 蚯蚓

- 65分：直接priority_queue
- 100分：结论：先切的蚯蚓分成的2块分别比后切的两块要大

  设先切x，切为⌊px⌋和x- ⌊px⌋　　(x>y)

  i秒后切（i秒时为y）现在为y+it，切为⌊p(y+it)⌋和y+it- ⌊p(y+it)⌋

  此时x的两块为⌊px⌋+it和x- ⌊px⌋+it

  显然 ⌊px⌋+it >⌊p(y+it)⌋

  而 y+it- ⌊p(y+it)⌋< y+it- ⌊py⌋< x- ⌊px⌋+it

P2168

Link

trie树

aa

aba

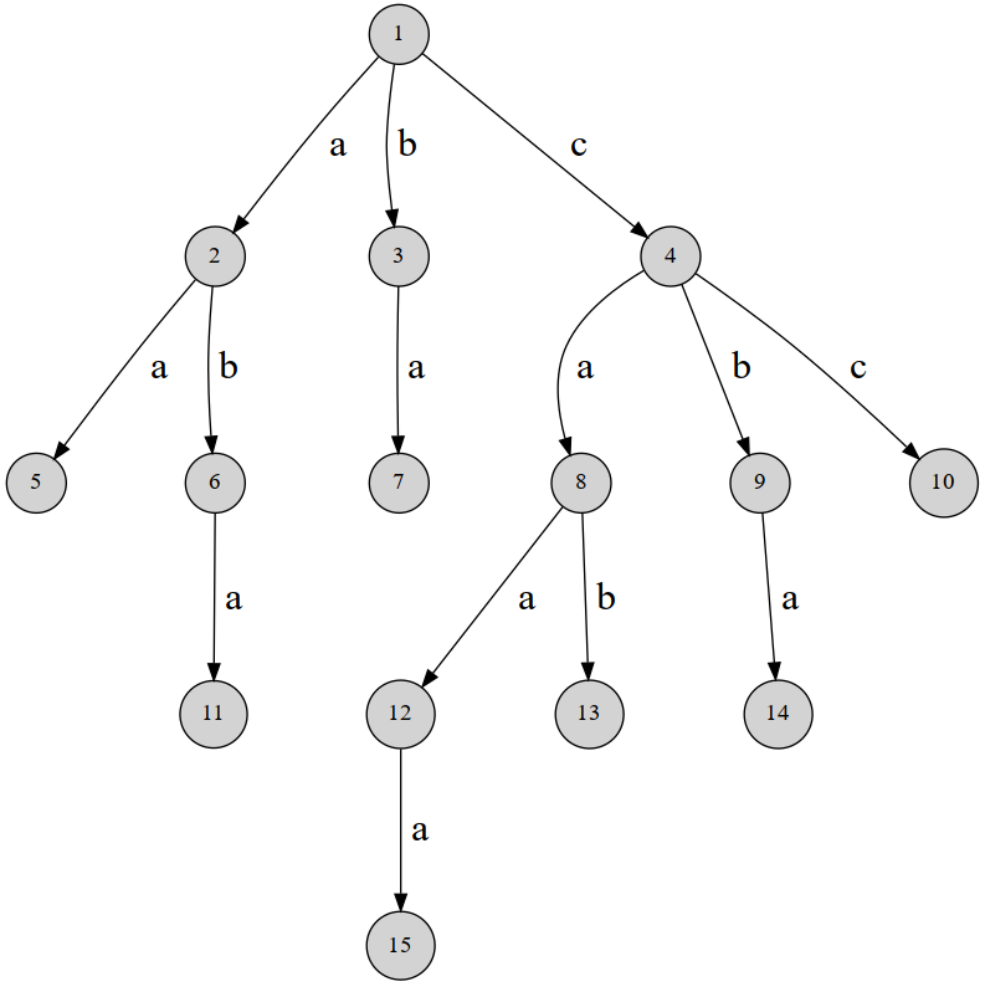
ba

caaa

cab

cba

cc



Trie

trie树

- 应用?
- 检索字符串（是否出现，出现次数计数）（相当于手动完成了map功能）
- 维护和前缀有关的一些信息，找两个字符串的公共前缀（trie上LCA）
- AC自动机

P2580

- Link

P5629

- Link

P4551

- Link


Thank You