# 基础数据结构---二叉堆

目录

[一、Supermarket](#_Toc2924_WPSOffice_Level1) [1](#_Toc2924_WPSOffice_Level1)

[二、 Sequence III](#_Toc8297_WPSOffice_Level1) [2](#_Toc8297_WPSOffice_Level1)

[三、 数据备份（双链表+堆）](#_Toc24731_WPSOffice_Level1) [4](#_Toc24731_WPSOffice_Level1)

[四、 合并果子 （二叉Huffman树、堆）](#_Toc21328_WPSOffice_Level1) [6](#_Toc21328_WPSOffice_Level1)

[五、 荷马史诗](#_Toc23808_WPSOffice_Level1) [6](#_Toc23808_WPSOffice_Level1)

#### 一、Supermarket

#include <bits/stdc++.h>

**using** **namespace** std;

**const** **int** maxn = 10005;

**int** n;

**struct** node{

**int** val , t;

}a[maxn];

**bool** cmp(node x , node y){

**return** x.t < y.t;

}

**int** main(){

    ios::sync\_with\_stdio(**false**);

    cin.tie(0);

**while**(cin >> n){

**for**(**int** i = 0 ; i < n ; ++ i) cin >> a[i].val >> a[i].t;

        //1.按照过期日期t排序

        sort(a , a + n , cmp);

        /\*以价值为值的小根堆 存放已经选择了的物品的价值

        同时它的size()也表示已经买了多少天\*/

        priority\_queue<**int** , vector<**int**> , greater<**int**> > pq;

**for**(**int** i = 0 ; i <n ; ++ i){

//t > size() 可以直接买

**if**(a[i].t > pq.size()) pq.push(a[i].val);

**else**{

                //若比堆中最小的大，替换它，以保持pq的size()小于等于a[i].t

**if**(a[i].val > pq.top()){

                    pq.pop();

                    pq.push(a[i].val);

                }

            }

        }

**long** **long** res = 0;

**while**(pq.size()){

            res += pq.top();

            pq.pop();

        }

        cout << res << '\n';

    }

}

#### Sequence III

**题目大意：**

有m组序列，每组有n个数。从每组中取一个数并求和，输出最小的n个和。

**解题思路：**

输入一组处理一组，先求出前两组数的n个最小和，再输入一组数，将这组数与前面的n个最小和一起处理，再求出这两组数的n个最小和…………直到m组数全部输入处理完毕。

处理过程如下：

①第一组数输入A数组中后升序排序，第二组数输入B数组后也升序排列

②算出A数组和B数组前n个最小的和，方法如下：

对于A[]、B[]有以下的值：

A1+B1 , A2+B1 , ... An + B1

A1+B2 , A2+B2 , ... An + B2

.

.

.

A1+Bn , A2+Bn , ... An + Bn

将A1+B1,A1+B2,...,A1+Bn这n个数加入小根堆中，选出最小的，堆顶就是当前的第一个最小的数，然后删去并加入该组的下一个数入堆，然后取n次堆顶就可取出n个最小的数。

例如开始堆中元素为A1+B1,A1+B2,...,A1+Bn，取出最小的数A1+B1之后，堆中pop(A1+B1)并加入A2+B1。

③遍历完毕后，将这n个最小元素赋给A数组，同时堆清空。

④输入下一组数到B数组，重复②，直到m组数输入处理完毕。

时间复杂度：O(mnlogn)

与**n路归并**思想和**分组**思想。

**代码：**

#pragma GCC optimize(3)

#include <bits/stdc++.h>

**using** **namespace** std;

**const** **int** maxn = 2005;

**int** a[maxn] ,tmp[maxn] , b[maxn];

**int** m , n;

**typedef** pair<**int**,**int**> pr; //<value,组号>

**void** merge(){

    priority\_queue<pr , vector<pr> , greater<pr> > heap;

**for**(**int** i = 0 ; i < n ; ++ i) heap.push({a[0] + b[i] , 0});

**for**(**int** i = 0 ; i < n ; ++ i){

        auto t = heap.top();

        heap.pop();

**int** val = t.first , pos = t.second;

        tmp[i] = val ;

        heap.push({val - a[pos] +  a[pos + 1] , pos + 1});

    }

**for**(**int** i = 0 ; i < n ; ++ i) a[i] = tmp[i];

}

**int** main(){

**int** T;

**scanf**("%d",&T);

**while**(T--){

**scanf**("%d %d",&m,&n);

**for**(**int** i = 0 ; i < n ; ++ i) **scanf**("%d",&a[i]);

        sort(a , a + n);

**for**(**int** i = 0 ; i < m - 1 ; ++ i){

**for**(**int** j = 0 ; j < n ; ++ j) **scanf**("%d",&b[j]);

            sort(b , b + n);

            merge();

        }

**for**(**int** i = 0 ; i < n ; ++ i) **printf**("%d ",a[i]);

**puts**("");

    }

**return** 0;

}

#### 数据备份（双链表+堆）

题意：有n个从左到右排好的点在数轴上，每个点都有一个坐标。现在需要从n个点中挑选k对点，每对点之间连一条直线，线的距离就是两个点之间的距离。（且k对点中无交集，每个点只属于一对点）提问k对线的总长度最小是多少。

代码：

#pragma GCC optimize(3)

#include <bits/stdc++.h>

**using** **namespace** std;

**const** **int** maxn = 100010;

**typedef** **long** **long** ll;

**typedef** pair<ll,**int**> pr; //<val,index>

**int** n , k;

**int** l[maxn] , r[maxn] ;

ll d[maxn];

**void** delete\_node(**int** p){ //从双链表中删去一个点

    r[l[p]] = r[p];

    l[r[p]] = l[p];

}

**int** main(){

**scanf**("%d %d",&n,&k);

**for**(**int** i = 0 ; i < n ; ++ i) **scanf**("%lld",&d[i]);

**for**(**int** i = n - 1 ; i ; i --) d[i] -= d[i - 1];

    set<pr> s;

    d[0] = d[n] = 1e15;

**for**(**int** i = 0 ; i <= n ; ++ i){

        l[i] = i - 1;

        r[i] = i + 1;

        s.insert({d[i],i});

    }

    ll res = 0;

**while**(k --){

        auto it = s.begin();

        ll v = it->first;

**int** p = it->second , left = l[p] , right = r[p];

        s.erase(it);

        s.erase({d[left],left}) , s.erase({d[right],right});

        delete\_node(left) , delete\_node(right);

        res += v;

        d[p] = d[left] + d[right] - d[p];

        s.insert({d[p] , p});

    }

**printf**("%lld\n",res);

**return** 0;

}

### Huffman树

对于k叉huffman树：

1. k = 2 正常贪心地从小根堆中选两个最小的合并
2. k > 2 若节点个数不符合 **(n - 1) mod (k - 1) = 0**的话，需要补一定数量的0，使得符合**(n - 1) mod (k - 1) = 0**的条件才能保证答案正确
3. **合并果子 （二叉Huffman树、堆）**

#include <bits/stdc++.h>

**using** **namespace** std;

**int** main(){

**int** n;

**scanf**("%d",&n);

**int** ans = 0;

    priority\_queue<**int**,vector<**int**>,greater<**int**> > pq;

**for**(**int** i = 0 , x; i < n ; ++ i) cin >> x , pq.push(x);

**while**(pq.size() != 1){

**int** x = pq.top();

        pq.pop();

**int** y = pq.top();

        pq.pop();

        ans += x + y;

        pq.push(x + y);

    }

**printf**("%d\n",ans);

**return** 0;

}

1. **荷马史诗**

#pragma GCC optimize(3)

#include <bits/stdc++.h>

**using** **namespace** std;

**const** **int** maxn = 100005;

**typedef** **long** **long** ll;

**typedef** pair<ll,**int**> pr;

pr a[maxn];

**int** main(){

**int** n , k;

**scanf**("%d%d",&n,&k);

    priority\_queue<pr, vector<pr> , greater<pr> > heap;

**for**(**int** i = 1 ; i <= n ; ++ i) {

**scanf**("%lld", &a[i].first);

        a[i].second = 0;

        heap.push(a[i]);

    }

**while**((n - 1) % (k - 1) != 0) {

        a[++ n].first = 0ll;

        a[n].second = 0;

        heap.push(a[n]);

    }

    ll ans = 0;

**while**(heap.size() != 1){

        ll val = 0 ;

**int** h = -1;

**for**(**int** i = 0 ; i < k ; ++ i){

            pr now = heap.top();

            heap.pop();

            h = max(h , now.second + 1);

            val += now.first;

        }

        ans += val;

        heap.push({val , h});

    }

**printf**("%lld\n%d\n",ans,heap.top().second);

**return** 0;

}