1018: Communication System

黄凯旋
数学科学学院

题目描述：

We have received an order from Pizoor Communications Inc. for a special communication system. The system consists of several devices. For each device, we are free to choose from several manufacturers. Same devices from two manufacturers differ in their maximum bandwidths and prices.

By overall bandwidth (B) we mean the minimum of the bandwidths of the chosen devices in the communication system and the total price (P) is the sum of the prices of all chosen devices. Our goal is to choose a manufacturer for each device to maximize B/P.

题目描述：

某公司希望从我们这里订购一种特殊的通讯系统。整个系统(system)包含若干个设备(devices)。对每个设备，我们可以从若干个制造商中选购。不同的制造商制造的同一种设备有不同的带宽(maximum bandwidths)和不同的价格(prices).

定义：总带宽 **B**(overall bandwidth)指所有设备的带宽的最小值；
总价格 **P**(total price)指所有设备的价格之和。

我们的目标是为每一个设备选取一个合适的制造商使得B/P最大。

输入：

第一行是测试样例数t. (1 <= t <= 10)
对每个测试样例，第一行包含一个正整数 n (1 ≤ n ≤ 100), 指通讯系统的总设备数。接着输入n行。第i行第一个数是 mi (1 ≤ mi ≤ 100), 指该设备可以选择的制造商个数，接着有mi个正整数对，每一个代表相应的制造商的带宽和价格。
Each test case starts with a line containing a single integer, the number of devices in the communication system, followed by n lines in the following format: the i-th line (1 ≤ i ≤ n) starts with mi (1 ≤ mi ≤ 100), the number of manufacturers for the i-th device, followed by mi pairs of positive integers in the same line, each indicating the bandwidth and the price of the device respectively, corresponding to a manufacturer.

输出：

对每一个测试样例输出最大的B/P比，保留三位小数。

样例输入：

> 1
> 3
> 3 100 25 150 35 80 25
> 2 120 80 155 40
> 2 100 100 120 110

样例输出：

> 0.649

Problem Restatement :

Given $(b_{i1}, p_{i1}), \cdots, (b_{in_i}, p_{in_i})$ for $i = 1, 2, \cdots, n$.

Choose $j_1, j_2, \cdots, j_n$

Maximize B/P

where

$$B = \min_{i=1,2,\cdots,n} \{b_{ij_i}\}$$

$$P = \sum_{i=1}^{n} p_{ij_i}$$

$$B = \min_{i=1,2,\cdots,n} \{b_{ij_i}\}$$

$$P = \sum_{i=1}^{n} p_{ij_i}$$

要最大化B/P, 我们希望B很大，P很小。

- 指定n个制造商后，B仅由最小的b_{ij_i}决定。
- 故：取定B的情况下最小化P，即将"非瓶颈"的设备换为价格最小者。

一种思路：

按带宽由小到大每次取定一个制造商作为“瓶颈”，其他设备选取比其带宽大的制造商中的价格最小者。

按总带宽由小到大顺次枚举，直到某个时刻某设备没有比它带宽还大的制造商，终止程序。

- 先将数据按带宽排序。指定某个设备，如何在它的所有制造商中找到价格最小的那个？
  - 价格可能是乱序的。（可能出现带宽又低、价格又高的伪劣品）数据无序时，寻找最小值复杂度为$O(N)$.
- 按价格作为次要关键词进行排序。
  - 不能解决问题！ 只能保证相同带宽的制造商价格升序排列，可能出现带宽又大价格又低的情况，这样每一次找最小值都需要遍历整个序列。

一种解决方法：

预处理：在某个设备的所有制造商中，如果某个制造商价格高而带宽低，则可以把它去掉。如果有多个价格相同的制造商，则选择带宽最大的那一个。
希望最后达到带宽严格升序、价格严格升序的情况。

- 利用`std::list`，删除复杂度为$O(1)$
- 先利用`std::list::sort`，将制造商按价格升序、带宽降序排列
    - 此时排在左边的价格小于等于右边，希望它的带宽比右边小。
- 再利用`std::list::unique`，将所有带宽小于左边的元素去掉。

（按带宽降序排列的好处：对于价格相同的情况，可以交由统一unique函数处理）

课后可以帮忙想一想： 如果在`std::list::sort`按带宽升序排列，则应该如何处理?

- 每个设备的制造商都如上处理后，全部都是按价格从小到大、带宽从小到大排序。
- 枚举的第一步： 每个设备都选排在最前面的制造商，找到最小带宽并计算总价格。
- 下一步，将带宽最小的元素去掉从list中去掉，重复上述过程。

优化思路

- 直接从n个设备中选取最小值：$O(n)$
  - 利用`std::priority_queue`,加入、删除元素复杂度为$O(\log(n))$.
- 直接累加计算n个设备的总价格：$O(n)$
  - 只需计算改变量，$O(1)$.

参考代码：

```
#include <bits/stdc++.h>
using namespace std;

struct Manu
{
  int Bandwith,Price;
  Manu(int _b,int _p):Bandwith(_b),Price(_p){ }
  friend bool operator < (const Manu& a, const Manu& b)
  { //先按价格升序，价格相同则按带宽降序
    if (a.Price==b.Price)
      return a.Bandwith>b.Bandwith;
    return a.Price<b.Price;
  }
};

bool Rmv(const Manu & a, const Manu & b)
{ //用于list::unique, 将带宽小于等于左边的删除
  if (a.Bandwith>=b.Bandwith) return true;
  else return false;
}
```

```
typedef list<Manu>::iterator IT;
struct iterid
{//存放list的迭代器、用于堆中
  IT it; int id;//代表出自哪个堆
  iterid(IT _it,int _id):it(_it),id(_id){}
  friend bool operator<(const iterid& a,const iterid& b)
  {
    return (a.it->Bandwith)>(b.it->Bandwith);
  }
};
```

```
int main()
{
  int TestCases,tmp,t1,t2;scanf("%d",&TestCases);
  while(TestCases--)
  {
    int n;scanf("%d",&n);
    list<Manu> bp[101];
    priority_queue<iterid> myheap;
    for (int i=0;i<n;++i)
    {
      scanf("%d",&tmp);
      for (int j=0;j<tmp;++j)
      {
        scanf("%d %d", &t1,&t2);
        bp[i].push_back(Manu(t1,t2));//放到list里
      }
      bp[i].sort();
      bp[i].unique(Rmv);//预处理
      myheap.push(iterid(bp[i].begin(),i));//首元素入堆
    }
```

```
double maxbp=0;
double currbp=0;
int currprice=0;
for (int i=0;i<n;++i)//第一步总归是要遍历的
  currprice+=bp[i].begin()->Price;
while(true)
{
  int currband=myheap.top().it->Bandwith;
  currbp=1.0*currband/currprice;//注意浮点运算.
  if (maxbp<currbp) maxbp=currbp;
  int nextid=myheap.top().id;
  //更新价格
  currprice-=bp[nextid].begin()->Price;
  myheap.pop();//O(log(n))
  bp[nextid].pop_front();
  if (bp[nextid].empty()) break;
  else
  {
    myheap.push(iterid(bp[nextid].begin(),nextid));
    //更新价格
    currprice+=bp[nextid].begin()->Price;
  }
}
printf("%.3lf\n",maxbp);
```

```
    }
    return 0;
}
```