

模拟费用流问题

laofu

陈江伦（清华大学交叉信息研究院）

January 27, 2019



自我介绍？

自我介绍？



自我介绍？

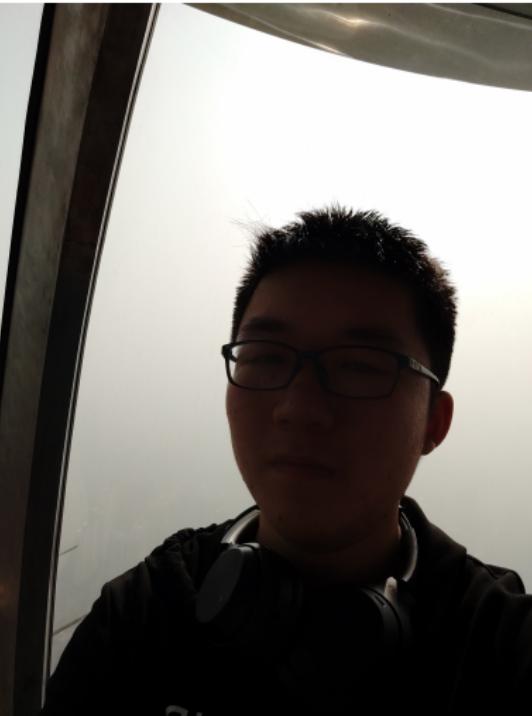


没有帽子更好

自我介绍？



没有帽子更好

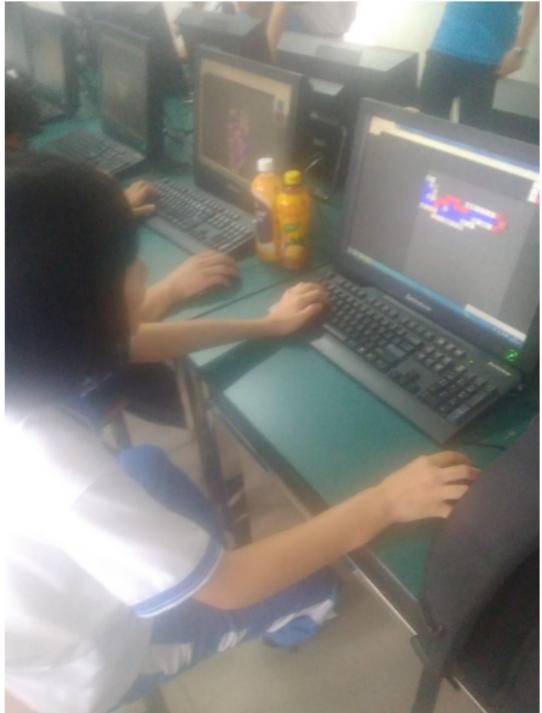


这节课讲什么？

这节课讲什么？

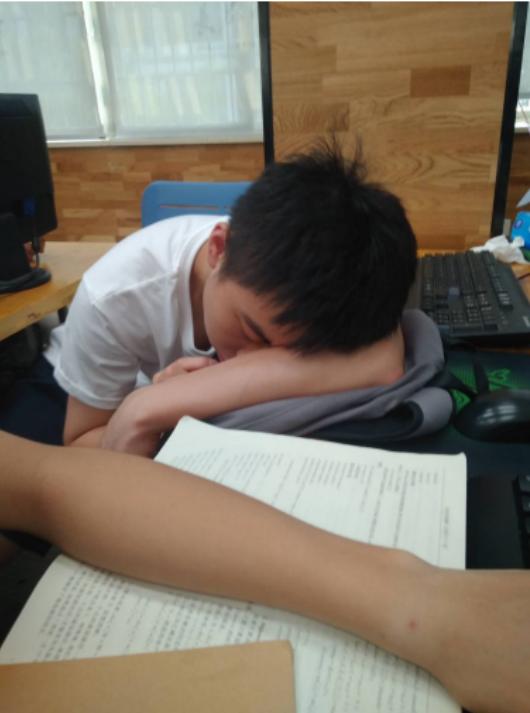
-讲费用流

-太简单了，我还是：



-讲大模拟

-太难了，我还是：



这节课讲什么？

费用流问题是算法竞赛中被广泛应用的问题。

费用流的考察一般主要是两个方面，一个方面是费用流的建模，另一方面是针对特殊图的费用流的优化。本课程主要介绍了第二个方面，即利用在特殊情况下图的特性，来优化费用流的效率。

费用流问题

给定网络，每条边有容量，有费用。

除了源汇之外，要求每个点流量守恒。

在最大化流量的前提下，最小化总费用。

给定网络，每条边有容量，有费用。

除了源汇之外，要求每个点流量守恒。

在最大化流量的前提下，最小化总费用。

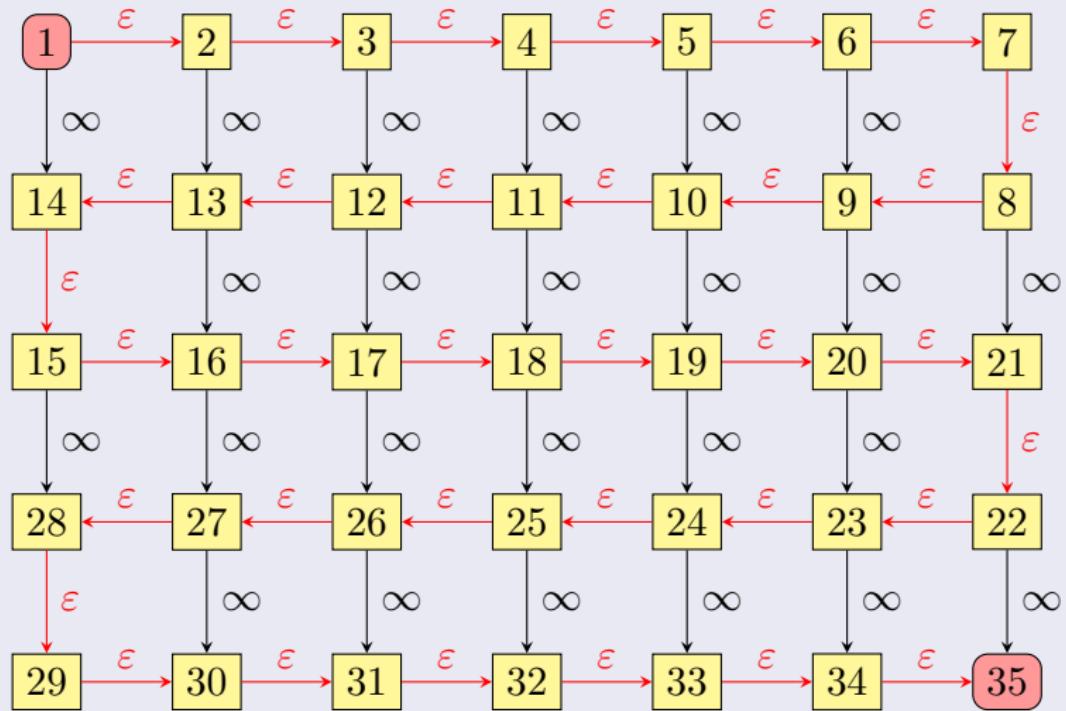
基础做法：EK

每次用 SPFA 算法找一条从 S 到 T 的最短路。

SPFA? 效率低下

SPFA? 效率低下

Hack 方法一



SPFA? 效率低下

Hack 方法二

- ① 生成一棵树高很高的树作为最短路树
- ② 对于剩下的边，随机端点 (a, b) ，边权为在 $[|d_a - d_b|, |d_a - d_b| + \varepsilon]$ (ε 为一个小常数) 内随机。

SPFA 的优化

LLL 优化

LLL 优化

做法：每次将入队结点距离和队内距离平均值比较，如果更大则插入至队尾。

LLL 优化

做法：每次将入队结点距离和队内距离平均值比较，如果更大则插入至队尾。

Hack：弄一个孤立点，只和 S 连一条权值为 ∞ 的边，这个优化就失效了。

SPFA 的优化

LLL 优化

做法：每次将入队结点距离和队内距离平均值比较，如果更大则插入至队尾。

Hack：弄一个孤立点，只和 S 连一条权值为 ∞ 的边，这个优化就失效了。

SLF 优化

LLL 优化

做法：每次将入队结点距离和队内距离平均值比较，如果更大则插入至队尾。

Hack：弄一个孤立点，只和 S 连一条权值为 ∞ 的边，这个优化就失效了。

SLF 优化

做法：设队首的元素为 i ，当往队列中加入一个元素 j 时，
 $d_j \leq d_i$ ，则把 j 加入到队首，否则加入到队尾。

LLL 优化

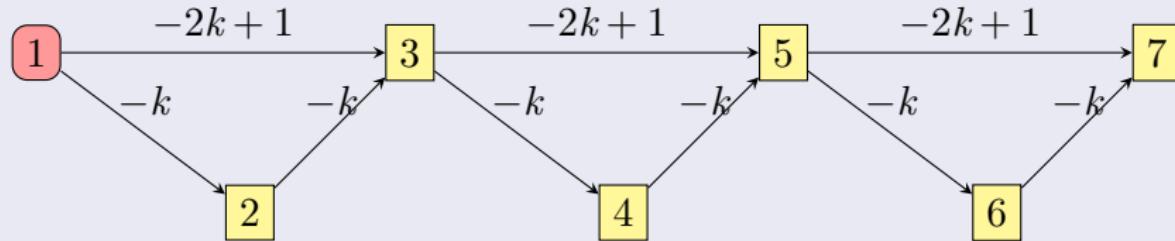
做法：每次将入队结点距离和队内距离平均值比较，如果更大则插入至队尾。

Hack：弄一个孤立点，只和 S 连一条权值为 ∞ 的边，这个优化就失效了。

SLF 优化

做法：设队首的元素为 i ，当往队列中加入一个元素 j 时， $d_j \leq d_i$ ，则把 j 加入到队首，否则加入到队尾。

Hack：



Johnson 最短路

Johnson 算法可以一次 SPFA 的代价把一张带负权的图转化成一张不带负权的图。

Johnson 算法可以一次 SPFA 的代价把一张带负权的图转化成一张不带负权的图。

步骤：

- ① 增加一个超级源 S_{Super} 。
- ② 用 SPFA 算法求出 S_{Super} 到每个点的距离 $h[1 \dots n]$ 。
- ③ 对于每条新边，赋值 $w'(u \rightarrow v) = w(u \rightarrow v) + h[u] - h[v]$
- ④ 在新图上跑最短路，原图的 $dis(u, v) =$ 新图上求出的 $dis'(u, v) - h[u] + h[v]$

这样可以在 $O(nm \log m)$ 的时间内负权图求出任意两点间的最短路。

Johnson 费用流

在处理完一条增广路之后，如何快速调整 h ？

在处理完一条增广路之后，如何快速调整 h ？

直接 $h'[k] = h[k] + dis'(S, k)$ 。

在处理完一条增广路之后，如何快速调整 h ？

直接 $h'[k] = h[k] + dis'(S, k)$ 。

Proof.

因为 $dis'(S, k)$ 表示的是从 S 到 k 的最短路，所以任取 u, v 都有：

$$dis'(S, u) + w'(u, v) \geq dis'(S, v)$$

$$dis'(S, u) + w(u, v) + h[u] - h[v] \geq dis'(S, v)$$

$$h'[u] + w(u, v) \geq h'[v]$$



同时，增加的反向弧费用为 0，亦满足非负条件。

EK 费用流：单路增广

ZKW 费用流：多路增广

做法：不断维护最短路图，然后在这张图上用多路增广运行最大流算法。

多路增广：把 BFS 改成在最短路图上 DFS。

动态维护最短路图：可以使用类似于 KM 的做法，维护每个点的顶标 $D[1 \dots n]$ 。当一组标号不存在流时，令 V 为从 S 出发沿最短路树能到达的点集， $d = \min_{i \in V, j \notin V, u(i,j) > 0} (D_i + c(i,j) - D_j)$ ，然后所有在 V 中的点标号增加 D 。

对比：在费用范围较小，流量较大，或者增广路径比较短的图中使用 ZKW 费用流有明显提速，反之则 ZKW 费用流可能比原始费用流更慢。

数轴上有 n 只老鼠和 m 个老鼠洞。

第 i 只老鼠的坐标为 $x[i]$, 第 j 个老鼠洞的坐标为 $y[j]$ 。

我们需要寻求一组匹配 $S = \{(a, b) | (a \text{ 为老鼠}, b \text{ 为洞})\}$ 。

Problem 1

简单的例子

Problem 1

简单的例子

description

每只老鼠只能往左走。

求所有老鼠都进洞的最小总代价（即行走的最小总距离）。

Problem 1

Problem 1

DP \Rightarrow Stack

Problem 1

DP \Rightarrow Stack

虽然很显然可以直接贪心

Problem 1

DP \Rightarrow Stack

虽然很显然可以直接贪心

问题等价于洞的权值为 $-y[i]$, 老鼠的权值为 $x[i]$, 求最小权匹配。
把所有的老鼠和洞按照坐标从小到大排序。

$f[i][j]$ 表示前 i 个老鼠和洞中, 有 j 个洞匹配了 i 之后的老鼠的最
小权代价。

如果 i 为老鼠, 则 $f[i][j] = f[i - 1][j + 1] + x[i]$ 。

如果 i 为洞, 则 $f[i][j] = \min(f[i - 1][j - 1] - y[i], f[i - 1][j])$ 。

注意到 $f[i][j]$ 一定是一些匹配再加上若干个孤立洞的坐标的相反数。而由于我们是按照坐标从小到大 DP, 所以 $-x[i]$ 是单调递减的。那么 $f[i - 1][j - 1] - y[i] < f[i - 1][j]$ 。

Problem 1

DP \Rightarrow Stack

虽然很显然可以直接贪心

问题等价于洞的权值为 $-y[i]$, 老鼠的权值为 $x[i]$, 求最小权匹配。
把所有的老鼠和洞按照坐标从小到大排序。

$f[i][j]$ 表示前 i 个老鼠和洞中, 有 j 个洞匹配了 i 之后的老鼠的最
小权代价。

如果 i 为老鼠, 则 $f[i][j] = f[i - 1][j + 1] + x[i]$ 。

如果 i 为洞, 则 $f[i][j] = \min(f[i - 1][j - 1] - y[i], f[i - 1][j])$ 。

注意到 $f[i][j]$ 一定是一些匹配再加上若干个孤立洞的坐标的相反
数。而由于我们是按照坐标从小到大 DP, 所以 $-x[i]$ 是单调递
减的。那么 $f[i - 1][j - 1] - y[i] < f[i - 1][j]$ 。

只需要把所有还未匹配的洞拿一个栈存储起来。每只老鼠一定是
匹配栈顶。

Problem 2

description

把上一个问题的老鼠只能往左走去掉。
求答案。

Problem 2

description

把上一个问题的老鼠只能往左走去掉。
求答案。

性质

老鼠 i 匹配洞 j 的代价为 $|x[i] - y[j]|$ 。

一个显然的性质：匹配不会相交



仍然可以按照从左到右的顺序来做。

Problem 2

DP \Rightarrow Stack

Problem 2

DP \Rightarrow Stack

与之前相比 $f[i][j]$ 的 j 可以为负数，表示还有 $-j$ 个老鼠需要匹配 i 之后的洞。

如果 i 为老鼠： $f[i][j] = f[i - 1][j + 1] + \begin{cases} x[i] & j \geq 0 \\ -x[i] & j < 0 \end{cases}$

如果 i 为洞：

$f[i][j] = \min(f[i - 1][j], f[i - 1][j - 1] + \begin{cases} y[i] & j \leq 0 \\ -y[i] & j > 0 \end{cases})$

Problem 2

DP \Rightarrow Stack

与之前相比 $f[i][j]$ 的 j 可以为负数，表示还有 $-j$ 个老鼠需要匹配 i 之后的洞。

如果 i 为老鼠： $f[i][j] = f[i - 1][j + 1] + \begin{cases} x[i] & j \geq 0 \\ -x[i] & j < 0 \end{cases}$

如果 i 为洞：

$f[i][j] = \min(f[i - 1][j], f[i - 1][j - 1] + \begin{cases} y[i] & j \leq 0 \\ -y[i] & j > 0 \end{cases})$

同样地 i 为洞且 $j \neq 0$ 时：

Problem 2

DP \Rightarrow Stack

与之前相比 $f[i][j]$ 的 j 可以为负数，表示还有 $-j$ 个老鼠需要匹配 i 之后的洞。

如果 i 为老鼠： $f[i][j] = f[i - 1][j + 1] + \begin{cases} x[i] & j \geq 0 \\ -x[i] & j < 0 \end{cases}$

如果 i 为洞：

$f[i][j] = \min(f[i - 1][j], f[i - 1][j - 1] + \begin{cases} y[i] & j \leq 0 \\ -y[i] & j > 0 \end{cases})$

同样地 i 为洞且 $j \neq 0$ 时：

$f[i][j] = f[i - 1][j - 1] + \begin{cases} y[i] & j < 0 \\ -y[i] & j > 0 \end{cases}$

Problem 2

DP \Rightarrow Stack

与之前相比 $f[i][j]$ 的 j 可以为负数，表示还有 $-j$ 个老鼠需要匹配 i 之后的洞。

如果 i 为老鼠： $f[i][j] = f[i - 1][j + 1] + \begin{cases} x[i] & j \geq 0 \\ -x[i] & j < 0 \end{cases}$

如果 i 为洞：

$f[i][j] = \min(f[i - 1][j], f[i - 1][j - 1] + \begin{cases} y[i] & j \leq 0 \\ -y[i] & j > 0 \end{cases})$

同样地 i 为洞且 $j \neq 0$ 时：

$f[i][j] = f[i - 1][j - 1] + \begin{cases} y[i] & j < 0 \\ -y[i] & j > 0 \end{cases}$

容易发现，这个 DP 有且仅有一个位置的值是需要决策的，其它位置都只是一个位移再整体加一个值。

Problem 2

DP \Rightarrow Stack

与之前相比 $f[i][j]$ 的 j 可以为负数，表示还有 $-j$ 个老鼠需要匹配 i 之后的洞。

如果 i 为老鼠： $f[i][j] = f[i - 1][j + 1] + \begin{cases} x[i] & j \geq 0 \\ -x[i] & j < 0 \end{cases}$

如果 i 为洞：

$f[i][j] = \min(f[i - 1][j], f[i - 1][j - 1] + \begin{cases} y[i] & j \leq 0 \\ -y[i] & j > 0 \end{cases})$

同样地 i 为洞且 $j \neq 0$ 时：

$f[i][j] = f[i - 1][j - 1] + \begin{cases} y[i] & j < 0 \\ -y[i] & j > 0 \end{cases}$

容易发现，这个 DP 有且仅有一个位置的值是需要决策的，其它位置都只是一个位移再整体加一个值。

可以用两个栈维护 DP 的 $j \geq 0$ 和 $j < 0$ 两个部分。并打上整体标记。复杂度 $O(\text{Sort}(n)) + O(n)$

Problem 3

跳伞

description

每只老鼠只能进入左边的洞。

每个洞 j 有个代价 $w[j]$, 老鼠 i 进入洞 j 的代价是
 $(x[i] - y[j] + w[j])$ 。

最大化总代价。 (不一定要让每只老鼠都进洞)

Problem 3

DP

Problem 3

DP

写出 DP 公式：

如果 i 为老鼠： $f[i][j] = \max(f[i - 1][j + 1] + x[i], f[i - 1][j])$ 。

如果 i 为洞： $f[i][j] = \max(f[i - 1][j - 1] - y[i] + w[i], f[i - 1][j])$ 。

Problem 3

DP

写出 DP 公式：

如果 i 为老鼠： $f[i][j] = \max(f[i - 1][j + 1] + x[i], f[i - 1][j])$ 。

如果 i 为洞： $f[i][j] = \max(f[i - 1][j - 1] - y[i] + w[i], f[i - 1][j])$ 。

当 i 为老鼠时，显然这个 DP 不需要决策。

当 i 为洞时，看起来需要决策。

Problem 3

Convex

Problem 3

Convex

Lemma

这个 DP 是凸的

Problem 3

Convex

Lemma

这个 DP 是凸的

每个时刻， $f[i][j]$ 的含义是当前还有 j 个洞没有匹配的带权匹配值。

如果 $f[i][j+1] - f[i][j] > f[i][j] - f[i][j-1]$ ，那么我们可以交换从 $f[i][j-1]$ 到 $f[i][j]$ 多选的那个洞和从 $f[i][j]$ 到 $f[i][j+1]$ 多选的那个洞。

定义 $d[i][j] = f[i][j] = f[i][j-1]$ ，则 $d[i]$ 是一个递减的序列。

Problem 3

Convex

Lemma

这个 DP 是凸的

每个时刻， $f[i][j]$ 的含义是当前还有 j 个洞没有匹配的带权匹配值。

如果 $f[i][j+1] - f[i][j] > f[i][j] - f[i][j-1]$ ，那么我们可以交换从 $f[i][j-1]$ 到 $f[i][j]$ 多选的那个洞和从 $f[i][j]$ 到 $f[i][j+1]$ 多选的那个洞。

定义 $d[i][j] = f[i][j] = f[i][j-1]$ ，则 $d[i]$ 是一个递减的序列。

再来看这个洞的转移式：

$f[i][j] = \max(f[i-1][j-1] - y[i] + w[i], f[i-1][j])$ 。为了保持凸性，相对于是把 $w[i] - y[i]$ 插入到了一个有序的 $d[i-1]$ 中，并继续保持有序。

Problem 3

Convex

Lemma

这个 DP 是凸的

每个时刻， $f[i][j]$ 的含义是当前还有 j 个洞没有匹配的带权匹配值。

如果 $f[i][j+1] - f[i][j] > f[i][j] - f[i][j-1]$ ，那么我们可以交换从 $f[i][j-1]$ 到 $f[i][j]$ 多选的那个洞和从 $f[i][j]$ 到 $f[i][j+1]$ 多选的那个洞。

定义 $d[i][j] = f[i][j] = f[i][j-1]$ ，则 $d[i]$ 是一个递减的序列。

再来看这个洞的转移式：

$f[i][j] = \max(f[i-1][j-1] - y[i] + w[i], f[i-1][j])$ 。为了保持凸性，相对于是把 $w[i] - y[i]$ 插入到了一个有序的 $d[i-1]$ 中，并继续保持有序。

从 $d[i-1]$ 到 $d[i]$ ，仅仅是插入了一个数。

Problem 3

Convex

再来看看对于老鼠的转移：

如果 i 为老鼠： $f[i][j] = \max(f[i - 1][j + 1] + x[i], f[i - 1][j])$ 。
设 i 之前洞的总个数为 T ，差分之后，就变成了

$$f[i][0] = f[i - 1][0] + d[i - 1][1] + x[i]$$

$$d[i][j] = \begin{cases} d[i - 1][j + 1] & j \in [1, T - 1] \\ -x[i] & j = T \end{cases}$$

Problem 3

优美的做法

Problem 3

优美的做法

那么，扫到 i 时，我们只需要维护 $f[i][0]$ 并保存所有的 $d[i]$ 就可以了。

每次仅仅需要查询的是 $d[i][1]$ ，利用凸性，就是查询最大值。

Problem 3

优美的做法

那么，扫到 i 时，我们只需要维护 $f[i][0]$ 并保存所有的 $d[i]$ 就可以了。

每次仅仅需要查询的是 $d[i][1]$ ，利用凸性，就是查询最大值。
维护一个数据结构，支持单点插入、求最大值、删除最大值。

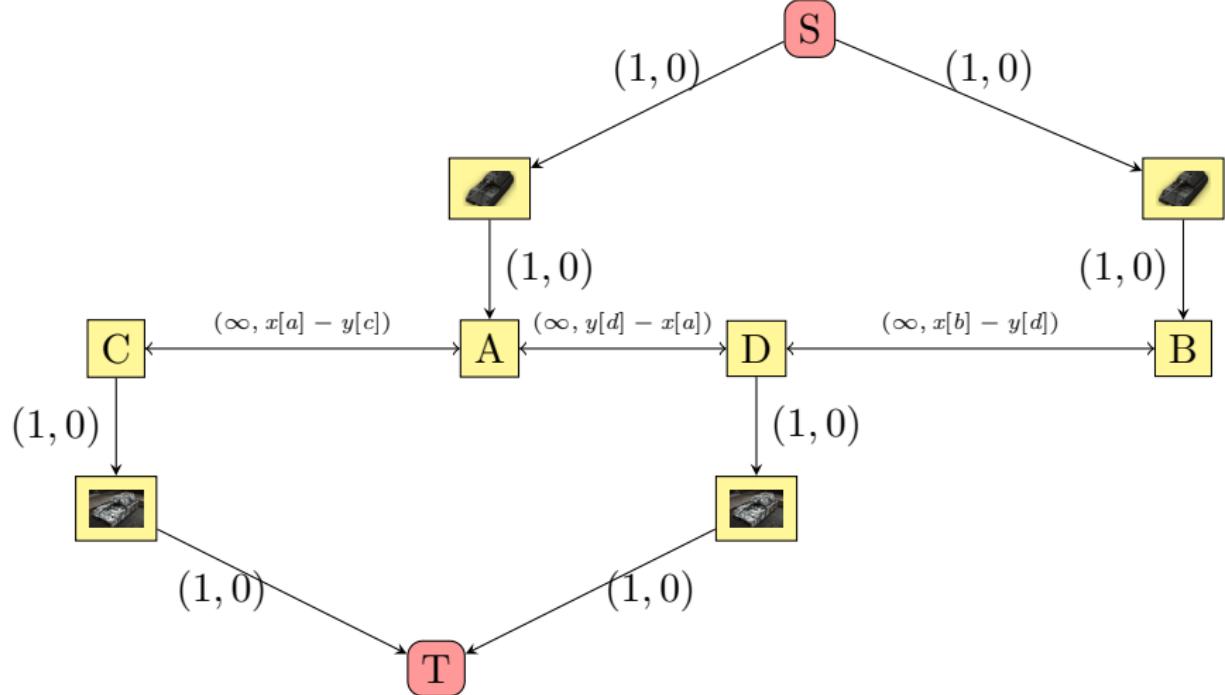


复杂度 $O(n \log n)$ 。

费用流模型

继续探讨 Problem 2

对于 Problem 2，我们可以建立一个费用流模型：



费用流模型

继续探讨 Problem 2

费用流模型

继续探讨 Problem 2

如何快速求费用流？

费用流模型

继续探讨 Problem 2

如何快速求费用流？

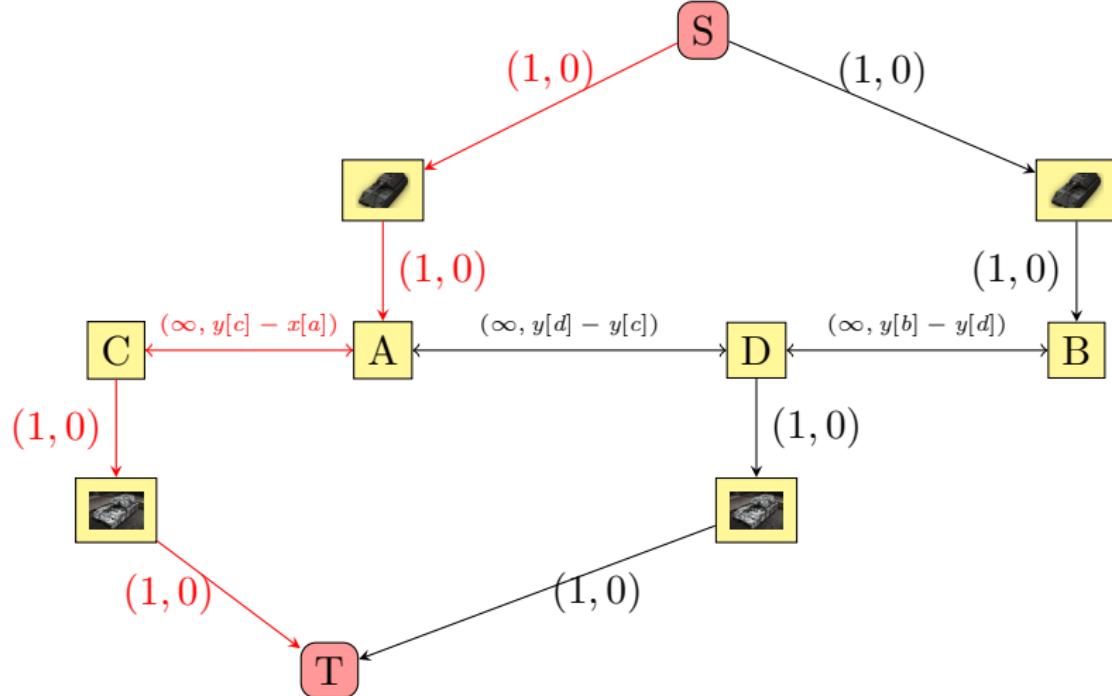
从左往右考虑每一个老鼠 i , 我们找到当前还剩下的洞中左边最近的 L_i 和右边最近的 R_i 。

然后分析老鼠到两个洞的距离。

费用流模型

Case 1

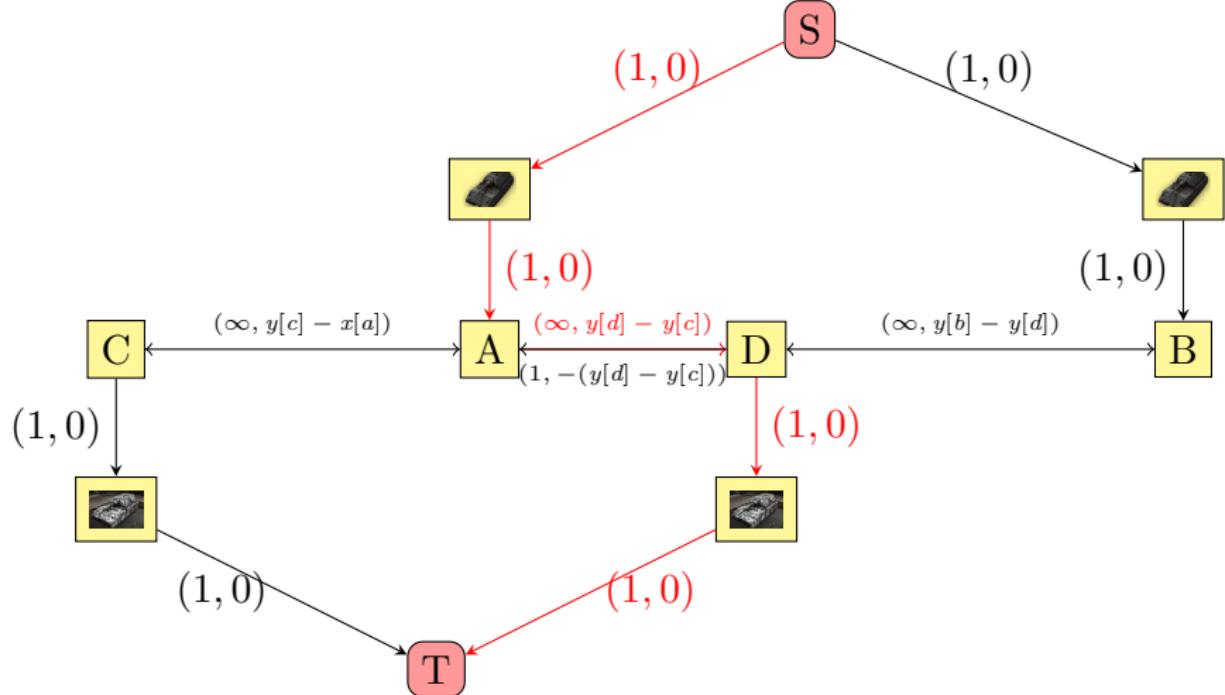
如果 $x[i] - y[L_i] \leq y[R_i] - x[i]$, 应该匹配左边的洞。



费用流模型

Case 2

如果 $x[i] - y[L_i] > y[R_i] - x[i]$, 应该匹配右边的洞。



费用流模型

动态维护反向弧

注意到，在 Case2 中，反向弧是有作用的。

费用流模型

动态维护反向弧

注意到，在 Case2 中，反向弧是有作用的。

当扫到坐标 q 时，如果 $p \rightarrow q$ 有一条反向弧，等价于把左边第一个洞的坐标往右移动 $2(q - p)$ 。

维护标记即可。复杂度 $O(\text{Sort}(n)) + O(n)$

堆的应用

回到 Problem 2

注意到，第二题的 DP 也是凸的，定义

$$d[i][j] = \begin{cases} f[i][j] - f[i][j-1] & j > 0 \\ f[i][j] - f[i][j+1] & j < 0 \end{cases}$$

堆的应用

回到 Problem 2

注意到，第二题的 DP 也是凸的，定义

$$d[i][j] = \begin{cases} f[i][j] - f[i][j-1] & j > 0 \\ f[i][j] - f[i][j+1] & j < 0 \end{cases}$$

如果 i 为老鼠：

$$f[i][0] = f[i-1][1] + d[i-1][1] + x[i]$$

$$d[i][j] = \begin{cases} d[i-1][j+1] & j > 0 \\ -x[i] - (d[i-1][1] + x[i]) & j = -1 \\ d[i-1][j+1] & j < -1 \end{cases}$$

如果 i 为洞：

$$f[i][0] = \min(f[i-1][0], f[i-1][0] + d[i-1][-1] + y[i])$$

堆的应用

回到 Problem 2

如果 $d[i-1][-1] + y[i] < 0$:

$$f[i][0] = f[i-1][0] + d[i-1][-1] + y[i]$$

$$d[i][j] = \begin{cases} -d[i-1][-1] - y[i] - y[i] & j = 1 \\ d[i-1][j-1] & j > 1 \\ d[i-1][j-1] & j < 0 \end{cases}$$

如果 $d[i-1][-1] + y[i] > 0$:

$$f[i][0] = f[i-1][0]$$

$$d[i][j] = \begin{cases} -y[i] & j = 1 \\ d[i-1][j-1] & j > 1 \\ d[i-1][j-1] & j < 0 \end{cases}$$

堆的应用

做法

维护两个 heap, $Q0$ 和 $Q1$ 。

Algorithm 1 Insertmouse($x[i]$)

- 1: $f_0 \leftarrow f_0 + Q1.top() + x[i];$
- 2: $Q0.push(-x[i] * 2 - Q1.top())$
- 3: $Q1.pop();$

Algorithm 1 Inserthole($y[i]$)

- 1: **if** $Q0.top() + y[i] < 0$ **then**
- 2: $f_0 \leftarrow f_0 + Q0.top() + y[i];$
- 3: $Q1.push(-Q0.top() - y[i] * 2);$
- 4: $Q0.pop();$
- 5: **else**
- 6: $Q1.push(-y[i]);$
- 7: **end if**

堆的应用 意义

堆的应用

意义

刚刚这个操作的意义?

堆的应用 意义

刚刚这个操作的意义？

相对于维护了两个增广集合，分别代表老鼠的增广和洞的增广。
当老鼠匹配完左边一个洞后，老鼠可以反悔改为匹配右边的洞。
当一个洞匹配完左边一个老鼠后，也可以反悔匹配右边的老鼠。

模拟费用流

两种做法对比

模拟费用流

两种做法对比

第一种做法：单向增广

第二种做法：双向增广

小 trick：我们建出来的图实际上是一个最小费用最大流问题。
而我们可以把每只老鼠带上 $-\infty$ 的权，转变为一个最小费用流问题。最后把答案加上 $\infty \times$ 老鼠个数。

Problem 4

加强

description

每个洞可以容纳多只老鼠，有一个容量上限。

Problem 4

Problem 4

神仙做法

我们做两遍贪心。

第一遍，我们强制每只老鼠只能往左，求一次答案 S_1 。

第二遍，我们强制每只老鼠只能往右，求一次答案 S_2 。

Problem 4

神仙做法

我们做两遍贪心。

第一遍，我们强制每只老鼠只能往左，求一次答案 S_1 。

第二遍，我们强制每只老鼠只能往右，求一次答案 S_2 。

Lemma

对于原问题，一定存在一组最优解满足这组解所选取的洞
 $\subseteq \min(S_1 + S_2, V)$ ， V 表示原来洞的集合。

Problem 4

神仙做法

我们做两遍贪心。

第一遍，我们强制每只老鼠只能往左，求一次答案 S_1 。

第二遍，我们强制每只老鼠只能往右，求一次答案 S_2 。

Lemma

对于原问题，一定存在一组最优解满足这组解所选取的洞

$$\subseteq \min(S_1 + S_2, V), V \text{ 表示原来洞的集合。}$$

Proof.

从第一遍贪心到原问题，对于每一个洞而言，从它右边出发到达它左边的老鼠不会变多。

从第二遍贪心到原问题，对于每一个洞而言，从它左边出发到达它右边的老鼠不会变多。 □

Problem 4

神仙做法

我们做两遍贪心。

第一遍，我们强制每只老鼠只能往左，求一次答案 S_1 。

第二遍，我们强制每只老鼠只能往右，求一次答案 S_2 。

Lemma

对于原问题，一定存在一组最优解满足这组解所选取的洞
 $\subseteq \min(S_1 + S_2, V)$ ， V 表示原来洞的集合。

Proof.

从第一遍贪心到原问题，对于每一个洞而言，从它右边出发到达它左边的老鼠不会变多。

从第二遍贪心到原问题，对于每一个洞而言，从它左边出发到达它右边的老鼠不会变多。 □

这样，洞的总容量降到了线性，就转化为了 Problem 2。

Problem 4

换一个做法

Problem 4

换一个做法

考虑之前的两类模拟费用流做法。

Problem 4

换一个做法

考虑之前的两类模拟费用流做法。

当加入一个老鼠时，我们的操作是不变的。

Problem 4

换一个做法

考虑之前的两类模拟费用流做法。

当加入一个老鼠时，我们的操作是不变的。

当加入一个洞时，我们可以往堆中丢一个 pair，分别表示增广的代价和容量。

注意到，每次增广都会使一只未匹配的老鼠匹配或者使一只往左跑的老鼠改为往右跑，所以堆操作的总次数依然是线性。

Problem 5

一道 IOI 题

description

每只老鼠可以任意次数分身，每只洞容量无限。

要求每只老鼠至少有一个分身进洞，每只洞至少进一只老鼠。

Problem 5

做法

Problem 5

做法

在费用流建图时，把每只老鼠拆成两类：

第一类：容量为 1，额外费用为 $-\infty$ 。

第二类：容量为 ∞ ，额外费用为 0。

仍然可以使用堆模拟费用流算法。

Problem 5

一些小变动

因为两侧都有了容量为 ∞ 的点，所以我们要重新分析一下复杂度。

如果是第一类点 a 匹配了左边某一个点 b ，则 b 不可能反悔。因为 a 是必须匹配的， b 一旦反悔， a 和 b 将同时连往 a 右边的点，匹配交叉，矛盾。

如果是第二类点 a 匹配了左边某一个点 b ，则 b 一定是第一类点。因为 b 是必须匹配的，所以 a 一旦反悔， a 和 b 将同时连往 a 右边的点，匹配交叉，矛盾。

所以，每次匹配完之后，只有一方可能反悔。增广总次数不超过 $2n$ 。

Problem 6

加强

description

在最基础的模型上加一个，每个洞有附加费用 $c[i]$ 。

Problem 6

一个大变动

在采用堆优化费用流模型的同时，发现一个问题：一个洞匹配了左边的一只老鼠之后，老鼠可能反悔，洞也有可能反悔。
如何保证复杂度？

Problem 6

一个大变动

在采用堆优化费用流模型的同时，发现一个问题：一个洞匹配了左边的一只老鼠之后，老鼠可能反悔，洞也有可能反悔。
如何保证复杂度？

NO PROBLEM! 如果老鼠和洞同时反悔，会发生匹配交叉，一定不优！

所以增广总次数仍然是线性。

Problem 7

一道 WF 题

description

模型变到了一棵树上，每棵树带边权。

Problem 7

一道 WF 题

description

模型变到了一棵树上，每棵树带边权。



World Finals | 2018 Beijing
acm icpc International Collegiate Programming Contest



Peking University
and CYSC



方正集团
FOUNDER
global sponsor

JETBRAINS
global sponsor

ACM ICPC World Finals 2018

final standings

| RANK | TEAM | SCORE | A | B | C | D | E | F | G | H | I | J | K |
|------|--|--------|----------------|---------------|----------------|----------------|---------------|----------------|----------------|----------------|---------------|---|---|
| 1 | Moscow State University <small>Northern Eurasia</small> | 9 1427 | 79 1 try | 79 1 try | 25 1 try | 296 4 tries | 47 4 tries | 140 2 tries | 41 2 tries | 224 3 tries | 55 1 try | | |
| 2 | Moscow Institute of Physics & Technology | 8 960 | 66 1 try | 17 1 try | 272 1 try | 229 2 tries | 55 1 try | | 91 1 try | 122 1 try | 48 3 tries | | |
| 3 | Peking University <small>Asia East</small> | 8 1190 | 102 2 tries | 41 3 tries | 203 2 tries | | 76 2 tries | 296 2 tries | 149 2 tries | 170 1 try | 33 2 tries | | |
| 4 | The University of Tokyo <small>Asia Pacific</small> | 8 1566 | 128 2 tries | 76 1 try | 281 4 tries | 298 7 tries | 65 5 tries | | 120 1 try | 216 4 tries | 42 1 try | | |
| 5 | Seoul National University | 7 719 | 105 2 tries | 16 1 try | | 262 5 tries | 53 1 try | | 74 1 try | 82 1 try | 27 1 try | | |
| 6 | University of New South Wales | 7 855 | 52 1 try | 31 1 try | 233 1 try | | 65 1 try | 275 2 tries | 123 6 tries | 16 2 tries | | | |
| 7 | Tsinghua University | 7 1002 | 105 1 try | 2 1 try | | 286 3 tries | 42 2 tries | 146 4 tries | 181 2 tries | 80 1 try | | | |
| | Chongqing Jiaotong University | 7 1016 | 99 1 try | 63 1 try | | 273 4 tries | 49 1 try | 130 2 tries | 175 1 try | 87 1 try | | | |

| Problem | A | B | C | D | E | F | G | H | I | J | K |
|-------------|-----|-----|---|----|----|-----|----|-----|-----|---|-----|
| Solved | 106 | 135 | 0 | 5 | 10 | 126 | 7 | 46 | 47 | 0 | 124 |
| Submissions | 262 | 247 | 4 | 15 | 91 | 462 | 75 | 278 | 229 | 1 | 254 |

Problem 7

简单做法

问题变到了树上，但是做法没有变化。
把堆改成可并堆即可。

Problem 8

加强再加强

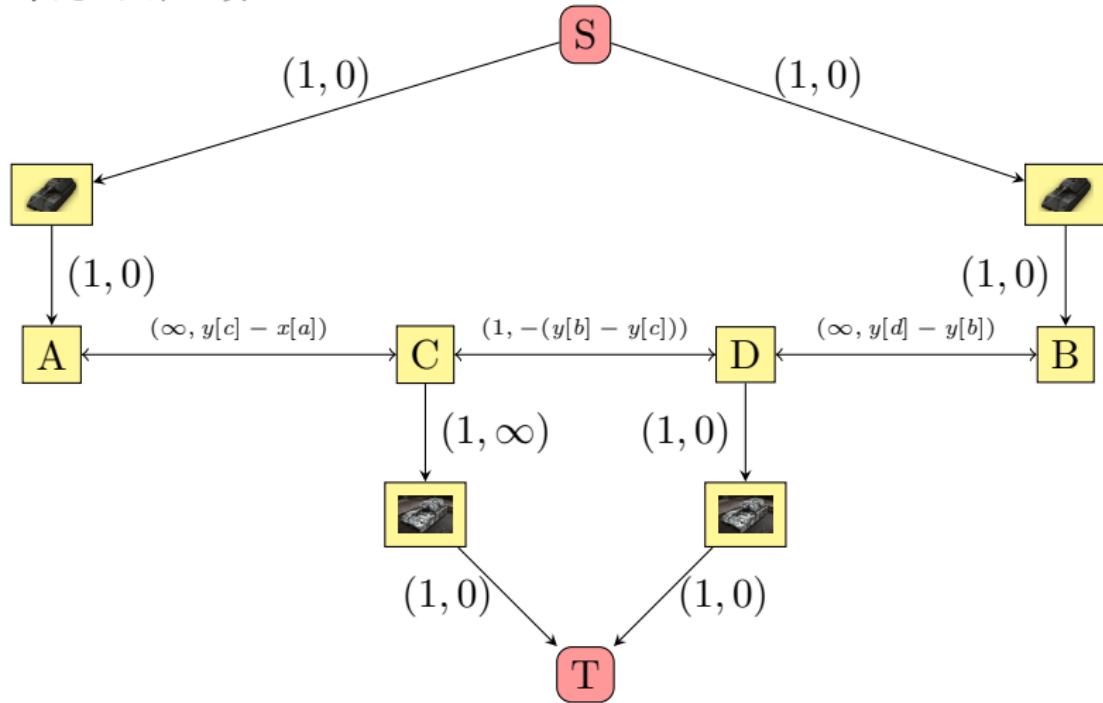
description

每个洞有容量，每个洞还有一个额外代价。

Problem 8

仍然用老方法？

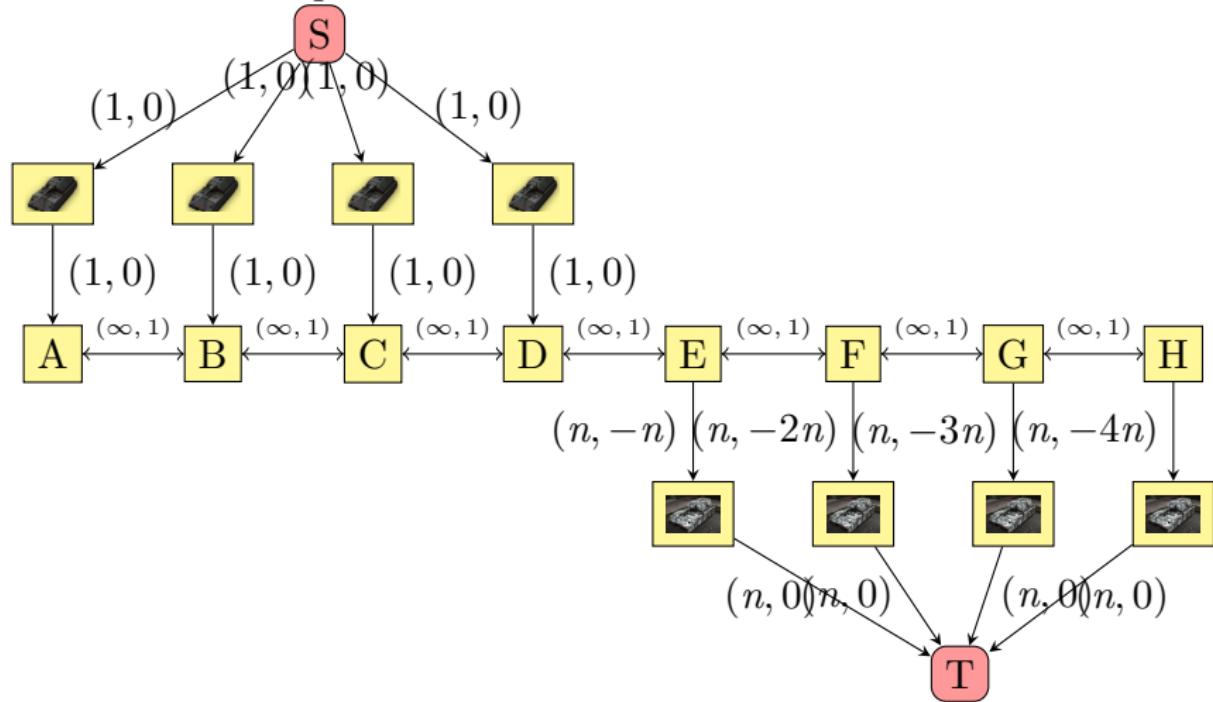
考虑之前的贪心？



Problem 8

仍然用老方法？

考虑之前那个放 pair 的费用流？



Problem 8

发现关键性质

考虑 Problem 6 的过程，我们维护了老鼠堆和洞堆。

Problem 8

发现关键性质

考虑 Problem 6 的过程，我们维护了老鼠堆和洞堆。

当遇到一只老鼠时，老鼠可能会先匹配之前的堆，然后往老鼠堆中丢一个这只老鼠反悔的操作。

这个过程放到本题，堆操作总次数依然是线性。

Problem 8

发现关键性质

考虑 Problem 6 的过程，我们维护了老鼠堆和洞堆。

当遇到一只老鼠时，老鼠可能会先匹配之前的堆，然后往老鼠堆中丢一个这只老鼠反悔的操作。

这个过程放到本题，堆操作总次数依然是线性。

当遇到一个洞时，洞可能会匹配一大堆老鼠，即从老鼠堆中取一大堆元素。然后洞和老鼠都有可能反悔，这样要同时往老鼠堆和洞堆中丢元素。



此时老鼠堆的操作次数爆炸啦。

但是注意到，对于老鼠 x, y 和洞 a, b 来说，假设坐标从左到右是 $x \rightarrow y \rightarrow a \rightarrow b$ 。那么，当老鼠 x 和 y 都匹配了洞 a 时，它们往老鼠堆中丢的元素都是 $-w_a - a$ 的坐标。也就是说，每次一个洞匹配了若干个老鼠之后，往老鼠堆中丢的元素都是等权的。这样我们就只需要在堆中对每个元素记录一下出现的次数，堆的操作总次数就变成了线性。

Problem 8

改进的贪心

我们做两遍贪心。

第一遍，我们强制每只老鼠只能往左，然后把所有老鼠分身一份，强制每只老鼠只能往右，求一次答案 S_1 。

第二遍，我们强制每只老鼠只能往右，然后把所有老鼠分身一份，强制每只老鼠只能往左，求一次答案 S_2 。

Problem 8

改进的贪心

我们做两遍贪心。

第一遍，我们强制每只老鼠只能往左，然后把所有老鼠分身一份，强制每只老鼠只能往右，求一次答案 S_1 。

第二遍，我们强制每只老鼠只能往右，然后把所有老鼠分身一份，强制每只老鼠只能往左，求一次答案 S_2 。

Lemma

对于原问题，一定存在一组最优解满足这组解所选取的洞
 $\subseteq \min(S_1 + S_2, V)$, V 是原来洞的集合。

Problem 8

改进的贪心

我们做两遍贪心。

第一遍，我们强制每只老鼠只能往左，然后把所有老鼠分身一份，强制每只老鼠只能往右，求一次答案 S_1 。

第二遍，我们强制每只老鼠只能往右，然后把所有老鼠分身一份，强制每只老鼠只能往左，求一次答案 S_2 。

Lemma

对于原问题，一定存在一组最优解满足这组解所选取的洞
 $\subseteq \min(S_1 + S_2, V)$, V 是原来洞的集合。

回到了 Problem 6。

Bonus Problem

遗憾的是，老夫并不能证明出这个 lemma 的正确性。

同时，也找不到这个做法的任何反例。

同学们可以思考，尝试证明它的正确性或是给出一组反例。最先给出证明或者给出反例的同学将获得厚礼：

价值 2000 日元的 IOI 官方指定纪念品 IA 亚克力板。



Problem 9

更复杂的问题

description

老鼠有分身，洞有容量，洞有额外代价

Problem 9

强行维护

把堆换成平衡树强行维护。

Problem 10

一道经典题

description

一个序列，要求选出 k 个不相交区间，使得这些区间的元素和最大。

Problem 10

线段树维护费用流。

Problem 11

问题再复杂

description

老鼠有分身，洞有容量，洞有额外代价。
中间的双向边容量不再是 ∞ 。

Problem 11

问题再复杂

不慌，拿几棵线段树维护一下流量就好了。

Problem 12

一道 NOI 题

description

有 n 种蔬菜，第 i 种蔬菜单价是 a_i ，数量有 c_i ，每天有 x_i 份会变质。每天可以买 m 个单位蔬菜。有 k 个询问，每次问买 p 天蔬菜能够获得的最大价值。

Problem 12

一道 NOI 题

按照天数从小到大维护费用流。
本题有比线段树更加优美的做法。

Problem 13

最后一道小清新题

description

有一棵 n 个点的树， m 个人站在树根。每条边有一个边权。现在每个人都可以任意行走，经过一条边就要付出对应边权的代价。问最小代价使得每条边至少被一个人经过。

$n \leq 10^5$, $m \leq 50$ 。

Problem 13

最后一道小清新题

直接跑普通费用流就可以过了。

Problem 13

最后一道小清新题

直接跑普通费用流就可以过了。

跑费用流实际上就可以用一个 DP 来实现。

感谢

- 感谢 CCF 提供交流的平台
- 感谢 UOJ 提供了放 (du) 题 (lit) 的平台

- UOJ 题解

谢谢大家

祝大家考试顺 (xiu) 利 (xian)

记得还有 Bonus Problem 哟！