知识精炼（一）

主讲人：邓哲也

NOIP 2016 Day1 换教室

对于刚上大学的牛牛来说，他面临的第一个问题是如何根据实际情况申请合适的课程。

在可以选择的课程中，有2n节课程安排在n个时间段上。在第i（$1 \le i \le n$）个时间段上，两节内容相同的课程同时在不同的地点进行，其中，牛牛预先被安排在教室c_i上课，而另一节课程在教室d_i进行。

在不提交任何申请的情况下，学生们需要按时间段的顺序依次完成所有的n节安排好的课程。如果学生想更换第i节课程的教室，则需要提出申请。若申请通过，学生就可以在第i个时间段去教室d_i上课，否则仍然在教室c_i上课。

NOIP 2016 Day1 换教室

由于更换教室的需求太多，申请不一定能获得通过。通过计算，牛牛发现申请更换第i节课程的教室时，申请被通过的概率是一个已知的实数ki，并且对于不同课程的申请，被通过的概率是互相独立的。

学校规定，所有的申请只能在学期开始前一次性提交，并且每个人只能选择至多m节课程进行申请。这意味着牛牛必须一次性决定是否申请更换每节课的教室，而不能根据某些课程的申请结果来决定其他课程是否申请；牛牛可以申请自己最希望更换教室的m门课程，也可以 不用完 这m个申请的机会，甚至可以一门课程都不申请。

NOIP 2016 Day1 换教室

因为不同的课程可能会被安排在不同的教室进行，所以牛牛需要利用课间时间从一间教室赶到另一间教室。

牛牛所在的大学有v个教室，有e条道路。每条道路连接两间教室，并且是可以 双向通行 的。由于道路的长度和拥堵程度不同，通过不同的道路耗费的体力可能会有所不同。当第i（1<= i <= n-1）节课结束后，牛牛就会从这节课的教室出发，选择一条耗费体力最少的 路径 前往下一节课的教室。

现在牛牛想知道，申请哪几门课程可以使他因在教室间移动耗费的体力值的总和的 期望值 最小，请你帮他求出这个最小值。

$n, m \leqslant 2000$，$v \leqslant 300$

NOIP 2016 Day1 换教室

首先考虑计算从 x 到 y 的耗费体力最小的路径。

显然是最短路问题。

因为最多只有 300 个点，可以方便的使用 Floyd 算法先用 $O(V^3)$ 的时间复杂度预处理出任意两点间的最短路。

NOIP 2016 Day1 换教室

如何计算耗费体力值的期望值？

只用对每条边考虑有多少概率会经过这条边。

设概率为 p，这条边耗费的体力值是 w

那么这条边对答案的贡献就是 pw。

最后只要把每条边的贡献加起来就是耗费体力值的总和的期望。

NOIP 2016 Day1 换教室

用 f[i][j][p] 来表示前 i 节课中用了 j 次换教室的机会的最小体力总和的期望。

p = 0 或 1，表示第 i 节课是否用了换教室的机会。

目的是定位此时牛牛在哪个教室。

p = 0 ➔ 没换，在 c[i] 教室

p = 1 ➔ 换了，在 d[i] 教室

NOIP 2016 Day1 换教室

对第 i 节课，考虑决策：换 or 不换。

不换：

牛牛需要赶到教室 c[i]

有两种可能

一种是上一节课换了：

有k[i-1]的概率从d[i-1]赶来

有(1-k[i-1])的概率从c[i-1]赶来

另一种是没换：

从c[i-1]赶来

NOIP 2016 Day1 换教室

对第 i 节课，考虑决策：换 or 不换。

不换：

```
f[i][j][0] = min(
      f[i-1][j][0] + dis[c[i-1]][c[i]],
      f[i-1][j][1] + k[i-1] * dis[d[i-1]][c[i]] + (1
- k[i-1]) * dis[c[i-1]][c[i]]
);
```

NOIP 2016 Day1 换教室

对第 i 节课，考虑决策：换 or 不换。

换：

牛牛有 k[i] 的概率需要赶到教室 d[i]，

有（1-k[i]）的概率需要赶到教室 c[i]

而上一节课也有两种可能：

上一节课换了：

有k[i-1]的概率从d[i-1]赶来

有(1-k[i-1])的概率从c[i-1]赶来

上一节课没换：从c[i-1]赶来

NOIP 2016 Day1 换教室

对第 i 节课，考虑决策：换 or 不换。

换：

```
f[i][j][1] = min(
        f[i-1][j][0] + k[i] * dis[c[i-1]][d[i]] + (1 - k[i]) *
dis[c[i-1]][c[i]],
        f[i-1][j][1]
            + (1-k[i]) * k[i-1] * dis[d[i-1]][c[i]]
            + k[i] * k[i-1] * dis[d[i-1]][d[i]]
            + (1-k[i-1]) * (1-k[i]) * dis[c[i-1]][c[i]]
            + (1-k[i-1]) * k[i] * dis[c[i-1]][d[i]]
);
```

NOIP 2016 Day1 换教室

最后的答案就是 min{f[n][i][0], f[n][i][1] | 0≤i≤m}

预处理最短路：

时间复杂度 $O(V^3)$

DP状态数 O(nm)，转移 O(1)，时间复杂度 O(nm)

由于 f[i] 由 f[i-1] 转移过来，可以使用滚动数组优化空间。

空间复杂度 O(m)

下节课再见