四边形不等式优化讲解（详解） - CSDN博客

累加器传送门：

http://blog.csdn.net/NOIAu/article/details/71775000

本篇博文意在详细讲解如下内容

F. 什么是四边形不等式

S. 四边形不等式优化如何证明

T. 怎么用四边形不等式优化

（好气啊，这篇博文我写了两遍，第一遍的没有保存就关了）
（感谢博客园的Staginner，他的博客对我有很大影响）
（感谢wys大佬亲自为我查了一部分内容的错）
（如果本文有什么错误的话，请向我提出，非常感谢）
这是他的博客：
http://www.cnblogs.com/staginner/archive/2012/03/12/2391925.html

---

引入：

在dp问题中，我们经常遇见这样的一类问题

他们的dp转移方程是这样的

dp[i][j]=min{dp[i][k]+dp[k+1][j]+cost[i][j]}

显然for一边i，for一边j，在算dp[i][j]的时候还得for一遍k，这是O（n^3）的复杂度，这样的复杂度在很多时候是不能接受的，如果dp转移方程已经设计好了，也无法再在dp方程上优化，我们怎么来提高计算效率呢？(关于O（n^2）复杂度的证明我放在最后)

对于（ a < b <= c< d ）

如果有f[a][c]+f[b][d]<=f[b][c]+f[a][d]

(可以理解为一句话，交叉小于包含，即交叉的两个区间，a到c和b到d的值满足小于等于包含的两个区间[bc包含于ad])
则说这个东西满足四边形不等式，当然这个东西可能是dp数组，也可以是其他数组，比如引入里提到的cost数组，表示的是i到j的花费（比如合并石子问题）

给出两个定理：

1、如果上述的w函数同时满足区间包含单调性和四边形不等式性质，那么函数dp也满足四边形不等式性质
我们再定义s(i,j)表示 dp(i,j) 取得最优值时对应的下标（即 i≤k≤j 时，k 处的 dp 值最大，则 s(i,j)=k此时有如下定理
2、假如dp(i,j)满足四边形不等式，那么s(i,j)单调，即 s(i,j)≤s(i,j+1)≤s(i+1,j+1)
如果不知道为什么，没有关系，反正后面都要证明

（这一部分如果没有完全懂没有关系，PART-THREE里会结合题目来推导一遍，所以有一小部分推导在这里先不写出来，这一部分尽量先说原理和方法）

壹：证明**cost**为凸（满足四边形不等式）

显然，当且仅当对于所有的i，j，均满足如下式子
(令i< i+1<= j< j+1)
cost[i][j]+cost[i+1][j+1]<=cost[i][j+1]+cost[i+1][j]
(利用PART ONE的交叉小于包含写出来的式子)
转移一下式子
即要证明对于所有的i，j，要使
cost[i][j]-cost[i+1][j]<=cost[i][j+1]-cost[i+1][j+1]
试想一个函数f（j）=cost[i][j]-cost[i+1][j]，那么要证明这个四边形不等式成立，也就是要证明这个函数f（j）单调递增嘛

贰：证明**dp**为凸（满足四边形不等式）

显然我们需要证明对于任意的i，j
i< i+1<= j< j+1
均满足
dp[i][j]+dp[i+1][j+1]<=dp[i+1][j]+dp[i][j+1]
这一步其实是建立在壹的上面的，可以用壹的结论来推导贰的结论
假设式子右边的dp[i+1][j]取得最优值的时候的k为x（之前不是要for一遍k来寻找最优时候的解嘛），右边式子里的dp[i][j+1]取得最优值的时候的k为y，这个时候把k=x带入左式的dp[i][j]中，把k=y带入左式的dp[i+1][j+1]中，然后寻找dp数组和cost数组的联系，把左边的式子配凑出右边的式子然后完成证明
（可能看不懂这句话是什么意思，不是很擅长表达，具体的证明方式大家可以结合PART-THREE里详细论证过程来学习，如果没懂就很不舒服的，现在就可以结合PART THREE看啦）

叁：证明决策单调（以找**min**值为例）

如果我们用s[i][j]表示dp[i][j]取得最优解的时候k的位置的话
那么我们要证明如下结论的成立性：
s[i][j-1]<=s[i][j]<=s[i+1][j]
对于s[i][j-1]<=s[i][j]来说，我们先令dp[i][j-1]取得最优解的时候的k值为y，然后令除了最优值以外的其他值可以为x，这里我们由于要讨论单调性，所以让x小于y，即x<=y<=j-1< j
然后利用我们在壹和贰里已经证明了的结论，来列一个关于dp数组的四边形不等式，然后在两边同时加上cost数组以求拼凑出如下的结论
dp[i][j-1] (k=x)+dp[i][j] (k=y)<=dp[i][j-1] (k=y)+dp[i][j] (k=x)
(这个看起来也有点奇怪，不过PART-THREE会讲具体是怎样操作的)
然后进行移项
dp[i][j-1] (k=x)-dp[i][j-1] (k=y)<=dp[i][j] (k=x)-dp[i][j] (k=y)
可以这样理解
我们之前是令dp[i][j-1]取得最优值的时候k=y，所以对于所有的x小于y，一定有dp[i][j-1]（k=x）>=dp[i][j-1]（k=y）（因为我们这里最优值是指最小值），所以很自然的，dp[i][j] (k=x)-dp[i][j] (k=y)也必然会大于等于零，也就是说，有如下结论
dp[i][j] (k=x)>=dp[i][j] (k=y)
什么意思呢，让dp[i][j-1]可以取得最小值的y，对于dp[i][j]的决策来说，可以使得所有小于y的x的值都没有y优，所以dp[i][j]的最优决策一定不会小于y，所以如果我们用s[i][j]表示dp[i][j]取得最优值的时候的k值，那么一定有
s[i][j-1]<=s[i][j]
而s[i][j]<=s[i+1][j]的证明是类似的，读者可以自己再推一遍

终于到了激动人心的详细证明过程和如何使用四边形不等式进行优化，亦可赛艇！

看例题来讲解总是好的，这里我们就举一个大家都做过的广为熟知的题

合并石子问题

现在有n堆石子，要将石子按一定顺序地合成一堆，规定如下，每次只能移动相邻的两堆石子，合并费用为新和成一堆石子的数量，求把n堆石子全部合并到一起所花的最少或者最大花费

很容易想到这样一个dp转移
dp[i][j]=min{dp[i][k]+dp[k+1][j]}+cost[i][j]
震惊！这不就是之前所讲的模型嘛？原来之前O（n^3）方的合并石子问题还可以优化（我太弱了）
首先明确一点，cost[i][j]表示把第i堆到第j堆的石子和到一起的最后一步的代价，显然，之前无论怎么合并，最后一步的代价都是一样的，所以我们可以先预处理出这个cost数组，他等于cnt[j]-cnt[i-1]，其中cnt数组是前缀和
for一遍i，for一遍j，每算一次dp[i][j]还要for一遍k，自然是O(n^3)方，现在我们来按照规则判断是否可以用四边形优化

第一步（壹）证明**cost**为凸

对于所有的i，j，令其满足i< i+1<=j< j+1
我们需要证明
cost[i][j]+cost[i+1][j+1]<=cost[i+1][j]+cost[i][j+1]
移项
cost[i][j]-cost[i+1][j]<=cost[i][j+1]-cost[i+1][j+1]
令f（j）=cost[i][j]-cost[i+1][j]
f（j）=cnt[j]-cnt[i-1]-(cnt[j]-cnt[i])
f（j）=cnt[i]-cnt[i-1]

都跟j无关了，自然一定满足四边形不等式（这个时候是直接等于了，但没有违反四边形不等式）

第二步（贰）证明**dp**为凸

要推导dp[i][j]的凸性，自然要满足对任意的i，j，令i< i+1<=j< j+1
有如下结论
dp[i][j]+dp[i+1][j+1]<=dp[i+1][j]+dp[i][j+1]
令dp[i+1][j]取得最优值的时候k=x
令dp[i][j+1]取得最优值的时候k=y
令x < =y(之后还要令x > y，这里不再赘述，读者如有兴趣可以自行推导，方式相似)
将k=x代入dp[i][j]，k=y代入dp[i+1][j+1]
左式=dp[i][x]+dp[x+1][j]+cost[i][j]+dp[i+1][y]+dp[y+1][j+1]+cost[i+1][j+1]①
而对于i< i+1<=j< j+1
由于已经在壹中证明了cost的凸性，所以
cost[i][j]+cost[i+1][j+1]<=cost[i+1][j]+cost[i][j+1]②
我们会发现这个不等式的左边在①式中出现过，所以把②式中的左式和右式替换一下可以得到如下结论
dp[i][x]+dp[x+1][j]+cost[i][j]+dp[i+1][y]+dp[y+1][j+1]+cost[i+1][j+1]
<=
dp[i][x]+dp[x+1][j+1]+cost[i][j+1]+dp[i+1][y]+dp[y+1][j]+cost[i+1][j]

即dp[i][j]+dp[i+1][j+1]<=dp[i][j+1]+dp[i+1][j]
证毕

第三步（叁）证明决策单调

现在我们已经证明了cost数组和dp数组的凸性，要证明决策单调以证明优化的正确性

即要证明s[i][j-1]<=s[i][j]<=s[i+1][j]

对于s[i][j-1]<=s[i][j]

令dp[i][j-1]取得最小值时的k=y，对于所有x≠y，令x<=y

可以有如下推导

∵x+1<=y+1<=j-1< j

四边形不等式有：

dp[x+1][j-1]+dp[y+1][j]<=dp[y+1][j-1]+dp[x+1][j]

在式子两边同时加上dp[i][x]+cost[i][j-1]+dp[i][y]+cost[i][j] 可以得到

dp[i][x]+dp[x+1][j-1]+cost[i][j-1]+dp[i][y]+dp[y+1][j]+cost[i][j]

<=

dp[i][x]+dp[x+1][j]+cost[i][j]+dp[i][y]+dp[y+1][j-1]+cost[i][j-1]

dp[i][j-1]+dp[i][j]<=dp[i][j]+dp[i][j-1]

(k=x).....(k=y).....(k=x).....(k=y)

移项

dp[i][j-1]-dp[i][j-1]<=dp[i][j]-dp[i][j]

(k=x).....(k=y).....(k=x).....(k=y)

由于我们是令k=y时dp[i][j-1]取得最小值，那么dp[i][j-1] (k=x)一定大于等于dp[i][j-1] (k=y)，所以左式大于零，所以右式也大于零，所以对于dp[i][j-1]可以取到最优值的y，所有小于它的值，对于dp[i][j]来说，都没有y优，所以最优决策一定不是小于y的，如果令s[i][j]表示dp[i][j]取得最优值的时候的k值，那么一定有

s[i][j-1]<=s[i][j]

证毕

对于不等式后半部分的证明类似，读者有兴趣可以自己再证明一次

代码如下

```
#include<iostream>
#include<cstdio>
#include<cstring>
#define MAXN 10000+10
using namespace std;

int dp[MAXN][MAXN],s[MAXN][MAXN],cnt[MAXN];
int n;

void init(){
    scanf("%d",&n);
    for(register int i=1;i<=n;i++) scanf("%d",&cnt[i]),cnt[i]=cnt[i-1]+cnt[i];
    for(register int i=1;i<=n;i++){
        dp[i][i]=0;s[i][i]=i;
    }
    for(register int i=n;i>=1;i--){
        for(register int j=i+1;j<=n;j++){
            int temp=0x7fffffff;
            int te;
            for(int k=s[i][j-1];k<=s[i+1][j];k++){
                if(temp>dp[i][k]+dp[k+1][j]+cnt[j]-cnt[i-1]){
```

```
                    temp=dp[i][k]+dp[k+1][j]+cnt[j]-cnt[i-1];
                    te=k;
                }
            }
            dp[i][j]=temp;
            s[i][j]=te;
        }
    }
}

void print(){
    cout<<dp[1][n]<<endl;
}

int main(){
    init();
    print();
}
```

- 1
- 2
- 3
- 4