# DP：四边形不等式优化

## **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)

# poj 1160 Post Office (区间DP)

**Post Office**

|  |  |  |
| --- | --- | --- |
| ****Time Limit:**** 1000MS |  | ****Memory Limit:**** 10000K |
| ****Total Submissions:**** 15966 |  | ****Accepted:**** 8671 |

**Description**

There is a straight highway with villages alongside the highway. The highway is represented as an integer axis, and the position of each village is identified with a single integer coordinate. There are no two villages in the same position. The distance between two positions is the absolute value of the difference of their integer coordinates.   
  
Post offices will be built in some, but not necessarily all of the villages. A village and the post office in it have the same position. For building the post offices, their positions should be chosen so that the total sum of all distances between each village and its nearest post office is minimum.   
  
You are to write a program which, given the positions of the villages and the number of post offices, computes the least possible sum of all distances between each village and its nearest post office.

**Input**

Your program is to read from standard input. The first line contains two integers: the first is the number of villages V, 1 <= V <= 300, and the second is the number of post offices P, 1 <= P <= 30, P <= V. The second line contains V integers in increasing order. These V integers are the positions of the villages. For each position X it holds that 1 <= X <= 10000.

**Output**

The first line contains one integer S, which is the sum of all distances between each village and its nearest post office.

**Sample Input**

10 5

1 2 3 6 7 9 11 22 44 50

**Sample Output**

9

题意：在v个村庄中建立p个邮局，求所有村庄到它最近的邮局的距离和，村庄在一条直线上，邮局建在村庄上。

思路：首先求出在连续的几个村庄上建立一个邮局的最短距离，用数组dis[i][j]表示在第i个村庄和第j个村庄之间建一个邮局的最短距。

dis[i][j]=dis[i][j-1]+x[j]-x[(i+j)/2]; （村庄位置为x[i]）

用数组dp[i][j]表示在前i个村庄中建立j个邮局的最小距离。即在前k（k<i）个村庄建立j-1个邮局，在k+1到j个村庄建立一个邮局。

dp[i][j]=min(dp[i][j],dp[k][j-1]+dis[k+1][i])

#include<stdio.h>

#include<math.h>

#include<string.h>

#include<stdlib.h>

#include<algorithm>

using namespace std;

#define N 305

const int inf=0x3fffffff;

int dp[N][35]; *//在前i个村庄中建立j个邮局的最小耗费*

int dis[N][N];*//dis[i][j]：第i个村庄到第j个村庄建一个邮局的最短距离*

int x[N]; *//村庄位置*

int main(){

int v,p,i,j,k;

while(scanf("%d%d",&v,&p)!=-1){

for(i=1;i<=v;i++)

scanf("%d",&x[i]);

*//memset(dis,0,sizeof(dis));*

for(i=1;i<=v;i++)

for(j=i+1;j<=v;j++)

dis[i][j]=dis[i][j-1]+x[j]-x[(i+j)/2];

for(i=1;i<=v;i++){

dp[i][i]=0; *//一个村庄一个邮局距离为零*

dp[i][1]=dis[1][i]; *//前i个村庄建立一个邮局*

}

for(j=2;j<=p;j++){

for(i=j+1;i<=v;i++){

dp[i][j]=inf;

for(k=j-1;k<i;k++)

dp[i][j]=min(dp[i][j],dp[k][j-1]+dis[k+1][i]);

}

}

printf("%d\n",dp[v][p]);

}

return 0;

}

# 四边形不等式优化

定义1：当决策代价函数w满足w[a, c]+w[b, d]<=w[a, d]+w[b, c](a<=b<=c<=d)时，称w****满足四边形不等式****。

定义2：当函数w满足w[b, c]<=w[a, d](a<=b<=c<=d)时，称w****关于区间包含关系单调****。

如果状态转移方程****dp[i, j] = min{dp[i, k-1]+dp[k, j]}+w[i, j](i<k<=j)****，****且w满足四边形不等式****，则有

定理1：上式dp满足四边形不等式。

定理2：令让dp[i, j]取最小值的k为K[i, j]，则有K[i, j-1]<=K[i, j]<=K[i+1, j]。

定理1证明：设k=K[b, c]，则

dp[a, c] + dp[b, d]

<= dp[a, k-1] + dp[k, c] + w[a, c] + dp[b, k-1] + dp[k, d] + w[b, d]

<= dp[a, k-1] + dp[k, d] + w[a, d] + dp[b, k-1] + dp[k, c] + w[b, c]

  = dp[a, d] + dp[b, c]

定理2证明：设k=K[i, j-1]，令i<k'<k<j-1<j，根据定理1，则有

dp[k', j-1] + dp[k, j] <= dp[k', j] + dp[k, j-1]

记S[k, i, j]=w[i, j] + dp[i, k-1] + dp[k, j]，不等式两侧同时加上

w[i, j-1] + w[i, j] + dp[i, k-1] + dp[i, k'-1]，可得

S[k', i, j-1] + S[k, i, j] <= S[k', i, j] + S[k, i, j-1]

根据k的定义，显然有S[k, i, j-1]<=S[k', i, j-1]，故有S[k, i, j]<=S[k', i, j]，所以k'不可能是K[i,j]，因此K[i, j]>=K[i, j-1]。

表示区间[i,j]取得最小值的位置不会小于区间[i,j-1]取得最小值的位置。

同理可得K[i, j]<=K[i+1, j]。

由于推出了最优决策K的单调性，从而可以减少每个状态转移的状态数，将复杂度从O(N^3)优化到O(N^2)。做法是：按照j-i划分阶段来求K[i, j]。显然根据j-i的长度有N个阶段，不妨设每个阶段枚举的长度（j-i）为d。求K[i, j]的复杂度是O(K[i+1, j]-K[i, j-1]+1)，则第d个阶段（即计算K[1, 1+d] 到K[n-d, n]）复杂度O(K[n-d+1, n] - K[1,d] +n-d)<=O(N)。故总时间复杂度为O（N^2）。

最终方程变为dp[i, j] = min{dp[i, k-1]+dp[k, j]}+w[i, j](K[i, j-1]<k<=K[i+1, j])。

伪代码：

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

K[i][i] = i;

dp[i][i] = p[i];

sum[i] = sum[i-1] + p[i];

}

for (int l=2;l<=l;l++) //枚举长度

for (int i=1;i<=n-l+1;i++) { //枚举首指针

int j = i+l-1; //尾指针

dp[i][j] = maxw;

for (int k=K[i][j-1];k<=K[i+1][j];k++)

if (dp[i][j] > dp[i][k-1]+dp[k][j]) {

dp[i][j] = dp[i][k-1]+dp[k][j];

K[i][j] = k;

}

dp[i][j] += sum[j]-sum[i-1];

}

//poj 1160 Post Office (区间DP)

#include<iostream>

#include<cstdio>

#include<algorithm>

#include<cstring>

#define N 1003

#define inf 1000000000

using namespace std;

int s[N][N],dp[N][N],w[N][N],n,m,dis[N];

int main()

{

freopen("a.in","r",stdin);

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

for (int i=1;i<=n;i++) scanf("%d",&dis[i]);

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

w[i][i]=0;

for (int j=i+1;j<=n;j++)

w[i][j]=w[i][j-1]+dis[j]-dis[(i+j)/2];

}

for (int i=1;i<=n;i++) dp[i][1]=w[1][i],s[i][1]=0;

for (int i=2;i<=m;i++) {

s[n+1][i]=n;

for (int j=n;j>i;j--) {

dp[j][i]=inf;

for (int k=s[j][i-1];k<=s[j+1][i];k++) {

int t=dp[k][i-1]+w[k+1][j];

if (t<dp[j][i]) dp[j][i]=t,s[j][i]=k;

}

}

}

printf("%d\n",dp[n][m]);

}