3-1（I）

设t[i][j][k]是完成前k件事情，可以在A处理器上不超过i且在B上不超过j。

则根据书上的例题算法，可得如下代码。

#include<iostream>

#include<cstring>

using namespace std;

int max(int a,int b)

{

return(a>b?a:b);

}

int min(int a,int b)

{

return (a<b?a:b);

}

int main()

{

int a[10],b[10];

int t[100][100][10];

int n;

cin>>n;

memset(t,0,sizeof(t));

int suma=0,sumb=0;

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

{

cin>>a[i];

suma+=a[i];

}

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

{

cin>>b[i];

sumb+=b[i];

}

t[0][0][0]=1;

for(int k=1;k<=n;k++)

for(int i=0;i<=suma;i++)

for(int j=0;j<=sumb;j++)

{

if(i>=a[k]&&t[i-a[k]][j][k-1])

t[i][j][k]=1;

if(j>=b[k]&&t[i][j-b[k]][k-1])

t[i][j][k]=1;

}

int ans=0x7FFFFFFF;//ans要是一个足够大的数，否则不行。

for(int i=0;i<=suma;i++)

for(int j=0;j<=sumb;j++)

if(t[i][j][n])

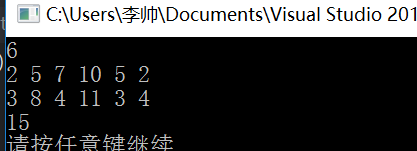
ans=min(ans,max(i,j));

cout<<ans<<endl;

system("pause");

return 0;

}



3-2（I）

首先定义这样一个函数——edit(i, j)，它表示第一个字符串的长度为i的子串到第二个字符串的长度为j的子串的编辑距离。

显然可以有如下动态规划公式：

if i == 0 且 j == 0，edit(i, j) = 0

if i == 0 且 j > 0，edit(i, j) = j

if i > 0 且j == 0，edit(i, j) = i

if i ≥ 1  且 j ≥ 1 ，edit(i, j) == min{ edit(i-1, j) + 1, edit(i, j-1) + 1, edit(i-1, j-1) + f(i, j) }，当第一个字符串的第i个字符不等于第二个字符串的第j个字符时，f(i, j) = 1；否则，f(i, j) = 0。

#include<iostream>

#include<string>

using namespace std;

int min(int a,int b)

{

return (a<b?a:b);

}

int edit(string a,string b)

{

int lena=a.length();

int lenb=b.length();

//int p[lena+1][lenb+1];

int \*\*p=new int\*[lena+1];

for(int i=0;i<lena+1;i++)

{

p[i]=new int[lenb+1];

}

for(int i=0;i<lena+1;i++)

p[i][0]=i;

for(int j=0;j<lenb+1;j++)

p[0][j]=j;

for(int i=1;i<=lena;i++)

for(int j=1;j<=lenb;j++)

{

int tmp=min(p[i-1][j]+1,p[i][j-1]+1);

int tmp1;

if(a[i-1]==b[j-1])

tmp1=p[i-1][j-1];

else

tmp1=p[i-1][j-1]+1;

p[i][j]=min(tmp,tmp1);

}

return p[lena][lenb];

}

int main()

{

string a,b;

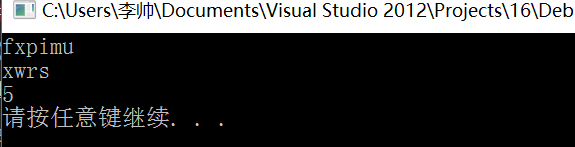
cin>>a>>b;

cout<<edit(a,b)<<endl;

system("pause");

return 0;

}



3-3(I)

算法思路：

 在此我们假设有n堆石子，一字排开，合并相邻两堆的石子，每合并两堆石子得到一个分数，最终合并后总分数最少的。

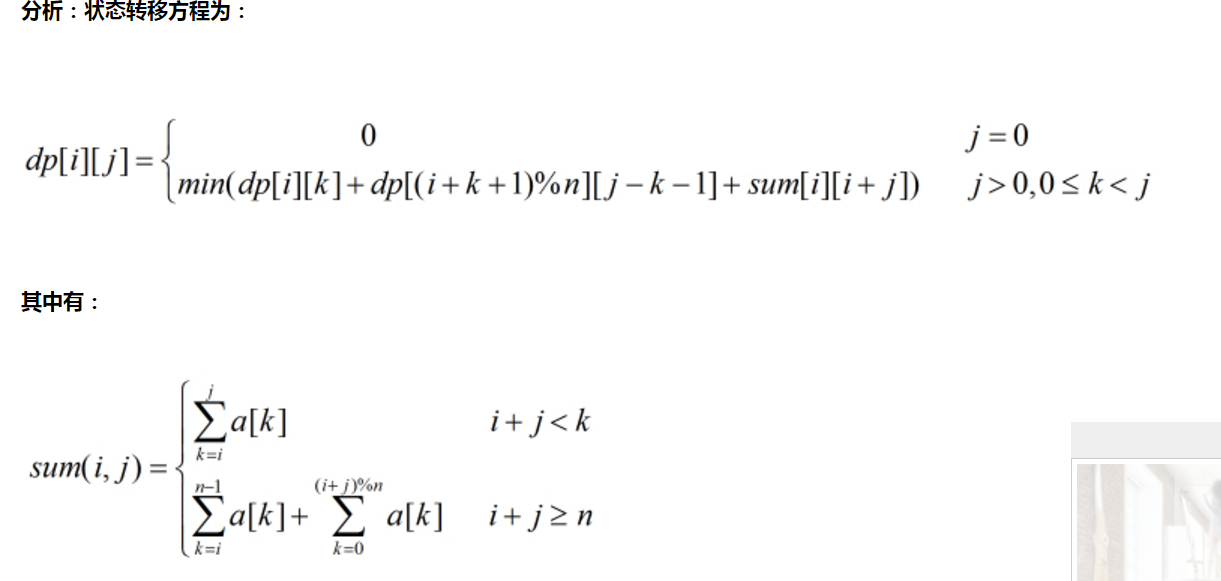
　　　我们设m(i,j)定义为第i堆石子到第j堆石子合并后的最少总分数。a(i)为第i堆石子得石子数量。

　　　当合并的石子堆为1堆时，很明显m(i,i)的分数为0;

　　   当合并的石子堆为2堆时，m(i,i+1)的分数为a(i)+a(i+1);

　　   当合并的石子堆为3堆时，m(i,i+2)的分数为MIN((m(i,i)+m(i+1,i+2)+sum(i,i+2)),(m(i,i+1)+m(i+2,i+2)+sum(i,i+2));

　　　后来又看到这个有直接的公式，于是换了一种方法。



解：

1./\*#include<iostream>

#include<cstring>

#include<vector>

#include<algorithm>

#define N 100

using namespace std;

int fmin(int\* p,int n)//求最小值

{

int m[N][N];

memset(m,-1,(n+1)\*(n+1));

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

m[i][i]=0;

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

{

m[i][i+1]=p[i]+p[i+1];

}

for(int r=3;r<=n;r++)

for(int i=1;i<=n-r+1;i++)

{

int j=i+r-1;

int sum=0;

for(int b=i;b<=j;b++)

sum+=p[b];

m[i][j]=m[i+1][j]+sum;

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

{

int t=m[i][k]+m[k+1][j]+sum;

if(t<m[i][j])

m[i][j]=t;

}

}

return m[1][n];

}

int fmax(int \*p,int n)//最大值

{

int m[N][N];

memset(m,-1,(n+1)\*(n+1));

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

m[i][i]=0;

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

{

m[i][i+1]=p[i]+p[i+1];

}

for(int r=3;r<=n;r++)

for(int i=1;i<=n-r+1;i++)

{

int j=i+r-1;

int sum=0;

for(int b=i;b<=j;b++)

sum+=p[b];

m[i][j]=m[i+1][j]+sum;

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

{

int t=m[i][k]+m[k+1][j]+sum;

if(t>m[i][j])

m[i][j]=t;

}

}

return m[1][n];

}

int MatrixChain\_max(int p[N],int n)

{

int m[N][N];

for(int x=1;x<=n;x++)

for(int z=1;z<=n;z++)

{

m[x][z]=-1;

}

int max=0;

//一个独自组合时

for(int g = 1;g<=n;g++) m[g][g]=0;

//两个两两组合时

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

{

int j=i+1;

m[i][j]=p[i]+p[j];

}

for(int r=3; r<=n;r++)

for(int i=1;i<=n-r+1;i++)

{

int j = i+r-1;

int sum=0;

for(int b=i;b<=j;b++)

sum+=p[b];

m[i][j] = m[i+1][j]+sum;

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

{

int t=m[i][k]+m[k+1][j]+sum;

if(t>m[i][j])

m[i][j] = t;

}

}

max=m[1][n];

return max;

}

int main()

{

int n;

cin>>n;

int\* t=new int[n+1];

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

cin>>t[i];

vector<int>maxn;

vector<int>minn;

maxn.push\_back(MatrixChain\_max(t,n));

minn.push\_back(fmin(t,n));

for(int i=2;i<=n;i++)

{

int start=t[1];

int z=2;

for(;z<=n;z++)

t[z-1]=t[z];

t[z]=start;

maxn.push\_back(MatrixChain\_max(t,n));

minn.push\_back(fmin(t,n));

}

cout<<\*min\_element(minn.begin(),minn.end())<<endl;

cout<<\*max\_element(maxn.begin(),maxn.end())<<endl;

system("pause");

return 0;

}

\*/

//2.

#include <iostream>

#include <string.h>

#include <stdio.h>

using namespace std;

const int INF = 1<<30;

const int N = 205;

int mins[N][N];

int maxs[N][N];

int sum[N],a[N];

int minval,maxval;

int n;

int getsum(int i,int j)

{

if(i+j >= n) return getsum(i,n-i-1) + getsum(0,(i+j)%n);

else return sum[i+j] - (i>0 ? sum[i-1]:0);

}

void Work(int a[],int n)

{

for(int i=0;i<n;i++)

mins[i][0] = maxs[i][0] = 0;

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

{

for(int i=0;i<n;i++)

{

mins[i][j] = INF;

maxs[i][j] = 0;

for(int k=0;k<j;k++)

{

mins[i][j] = min(mins[i][j],mins[i][k] + mins[(i+k+1)%n][j-k-1] + getsum(i,j));

maxs[i][j] = max(maxs[i][j],maxs[i][k] + maxs[(i+k+1)%n][j-k-1] + getsum(i,j));

}

}

}

minval = mins[0][n-1];

maxval = maxs[0][n-1];

for(int i=0;i<n;i++)

{

minval = min(minval,mins[i][n-1]);

maxval = max(maxval,maxs[i][n-1]);

}

}

int main()

{

while(scanf\_s("%d",&n)!=EOF)

{

for(int i=0;i<n;i++)

scanf\_s("%d",&a[i]);

sum[0] = a[0];

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

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

Work(a,n);

printf("%d %d\n",minval,maxval);

}

return 0;

}

