**关竣佑DP笔记**

**背包问题：**

**01背包：**朴素型

#include<bits/stdc++.h>//hdu2602 01背包 滚动数组

using namespace std;

struct bone

{

int val;//价值

int vol;//体积

}bone[1011];

int T,N,V;

int dp[1011];//总体积j

int ans()

{

memset(dp,0,sizeof(dp));

for(int i=1;i<=N;i++) //**注意此处从1开始遍历**

{

for(int j=V;j>=bone[i].vol;j--)//**注意到bone[i].vol停止 注意是逆序遍历**

{

dp[j]=max(dp[j],dp[j-bone[i].vol]+bone[i].val);

}

}

return dp[V];

}

int main()

{

cin>>T;

while(T--)

{

cin>>N>>V;

for(int i=1;i<=N;i++) cin>>bone[i].val;

for(int i=1;i<=N;i++) cin>>bone[i].vol;

cout<<ans()<<endl;

}

return 0;

}

概率型：

**P(成功的概率)=1-P(所有不成功概率的积)**

**每次尝试：Pi(成功)=1-pi(不成功)**

#include<bits/stdc++.h>//hdu1203 01背包

using namespace std;

int n,m;

int a[10010];//每个学校需要的钱

double b[10010];

double dp[10010][10010];

double solve()

{

**for(int i=0;i<=m;i++)**

**dp[0][i]=1;**

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

**dp[i][0]=1;**

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

{

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

{

if(i>a[i])

**dp[i][j]=min(dp[i][j-1],dp[i-a[i]][j-1]\*(1-b[i]));**

}

}

**return** **1.0-dp[n][m];**

}

int main()

{

cin>>n>>m;

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

cin>>a[i]>>b[i];

printf("%.1lf",solve()\*100);

cout<<"%"<<endl;

return 0;

}

#include<bits/stdc++.h>//hdu2955 01背包

#include<cmath>

using namespace std;

int N;

int n,val[110];

double v,w[110];

int sum=0;

double dp[110];//抢劫前i个银行，抢劫j元成功的概率

int solve()

{

dp[1]=1.0;

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

{

for(int j=sum;j>=val[i];j++)//注意滚动数组优化是逆序

{

**dp[j]=max(dp[j],dp[j-val[i]]\*(1.0-w[i]));**

}

}

for(int j=sum;j>=0;j--)

{

if(dp[j]>1.0-v)

return j;

}

}

int main()

{

int ans[110];

cin>>N;

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

{

sum=0;

memset(dp,0,sizeof(dp));

cin>>v>>n;

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

{

cin>>val[i]>>w[i];

sum+=val[i];

}

ans[i]=solve();

}

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

cout<<ans[i]<<endl;

return 0;

}

**完全背包：**

**朴素板子题：**

#include<bits/stdc++.h>//hdu2844多重背包

using namespace std;

int n,m;

int v[110],c[110];

int dp[110][110];

int vis[100010];

int solve()

{

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

{

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

{

**for(int k=0;k<=j/v[i]&&k<=c[i];k++)**

{

**dp[i][j]=max(dp[i][j],dp[i-1][j-k\*v[i]]+v[i]);**

}

}

}

return dp[n][m];

}

int main()

{

while(cin>>n>>m)

{

if(n==0&&m==0) break;

memset(dp,0,sizeof(dp));

memset(vis,0,sizeof(vis));

cout<<solve()<<endl;

}

return 0;

}

**动态数组优化：**

#include<bits/stdc++.h>//完全背包 洛谷疯狂的采药

using namespace std;

int t,m;

long long dp[10000010],a[10100],b[10100];//a是时间,b是价值

int main()

{

cin>>t>>m;

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

cin>>a[i]>>b[i];

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

{

**for(int j=1;j<=t;j++)//**  **注意顺序遍历**

{

if(j-a[i]>=0)

**dp[j]=max(dp[j],dp[j-a[i]]+b[i]);//这里和01背包一样**

else

**dp[j]=dp[j];**

}

}

cout<<dp[t];

return 0;

}

**有限制的完全背包：**

#include<bits/stdc++.h>//hdu2159 完全背包，动态数组

using namespace std;

#define mem(a,b) memset(a,b,sizeof(a))

const int maxn=150;

int n,m,k,s;

int a[maxn],b[maxn];

int dp[maxn][maxn];//dp[i][j]为所需耐力值为i杀怪数为j时所获得的 最大经验值

int solve()

{

mem(dp,0);

for(int k1=1;k1<=k;k1++)//遍历怪的种类

{

for(int i=b[k1];i<=m;i++)//耐力值

{

for(int j=1;j<=s;j++)//杀怪数

{

**dp[i][j]=max(dp[i][j],dp[i-b[k1]][j-1]+a[k1])**;//滚动数组

}

}

}

**for(int i=1;i<=m;i++)//都是从小到大遍历**

**{**

**for(int j=1;j<=s;j++)**

**if(dp[i][j]>=n)**

**return m-i;//返回最大忍耐值**

**}**

return -1;

}

int main()

{

while(scanf("%d%d%d%d",&n,&m,&k,&s)!=EOF)

{

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

{

cin>>a[i]>>b[i];

}

cout<<solve()<<endl;

}

return 0;

}

**多重背包：**

**朴素版：**

#include<bits/stdc++.h>//多重背包朴素

using namespace std;

int v[510],w[510],s[510];//每种价格，价值，数量

int dp[6100]

;

int main()

{

int n,m;//物品种数，现有钱数

cin>>n>>m;

for(int i=1;i<=n;i++) cin>>v[i]>>w[i]>>s[i];

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

{

for(int j=m;j>=1;j--)

{

**for(int k=0;k<=s[i]&&j>=k\*v[i];k++)**

**{**

**dp[j]=max(dp[j],dp[j-k\*v[i]]+k\*w[i]);**

//增加i个物品拿k个的循环

}

}

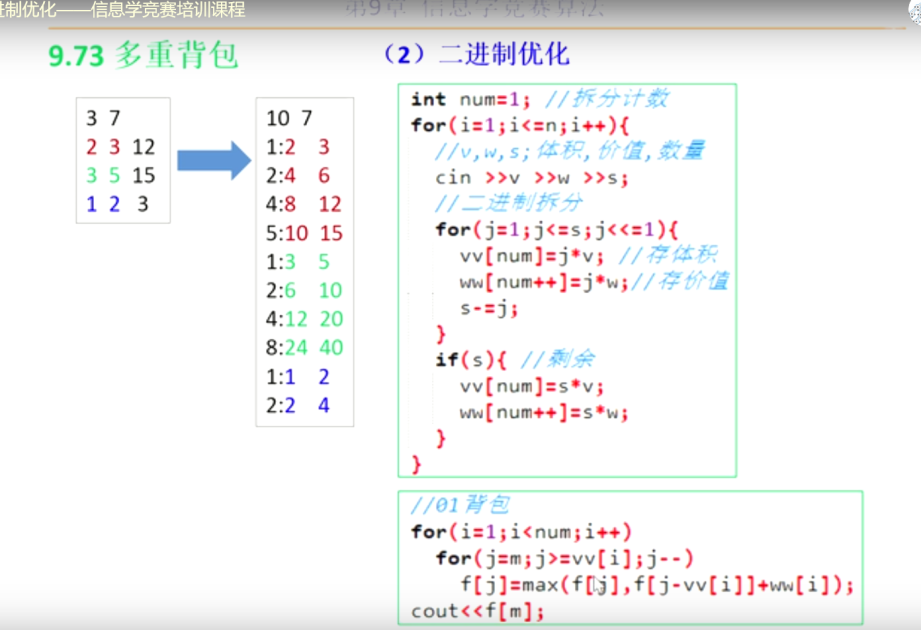
}

cout<<dp[m]<<endl;

return 0;

}

**二进制优化为01背包模板：**

****

#include<bits/stdc++.h>//多重背包二进制优化为01背包模板

using namespace std;

int vv[1000],ww[1000],n,m;

int dp[10000];

int v,w,s;//体积价值数量

int main()

{

int num=1;

cin>>n>>m;//总种类，容量

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

{

cin>>v>>w>>s;

for(int j=1;j<=s;j<<=1) //j<<=1是j乘以二再赋值给j

{

vv[num]=j\*v;

ww[num++]=j\*w;

s-=j;

}

if(s)//如果还有剩余份数

{

vv[num]=s\*v;

ww[num++]=s\*w;

}

}

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

{

for(int j=m;j>=vv[i];j--)

dp[j]=max(dp[j],dp[j-vv[i]]+ww[i]);

}

cout<<dp[m]<<endl;

return 0;

}

**区间DP：**

**加法朴素型区间DP：**

#include<bits/stdc++.h>//**石子合并**

using namespace std;

int n;

int a[255];

int sum[255];

int dp[255][255];

int solve()

{

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

{

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

{

**int j=i+len-1;**

**if(len==1) dp[i][j]=0;**

else

{

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

{

**int tmp=dp[i][k]+dp[k+1][j]+(sum[j]-sum[i-1]);**

**dp[i][j]=min(dp[i][j],tmp);**

}

}

}

}

return dp[1][n];

}

int main()

{

cin>>n;

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

{

cin>>a[i];

**sum[i]=sum[i-1]+a[i];//前缀和**

}

**memset(dp,0x7f7f7f,sizeof(dp));//记得初始化为大数**

cout<<solve()<<endl;

return 0;

}

**变式1：**

#include<bits/stdc++.h>//hdu5115 区间DP 说狼

using namespace std;

int n,T;

int a[210],dp[210][210],b[210];

int inf=0x7f7f7f;

int solve()

{

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

{

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

{

int j=i+len-1;

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

**int tmp=dp[i][k-1]+dp[k+1][j]+a[k]+b[k-1]+b[k+1];**

dp[i][j]=min(dp[i][j],tmp);}

}

}

return dp[1][n];

}

int main()

{

cin>>T;

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

{

cin>>n;

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

cin>>a[j];

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

cin>>b[j];

memset(dp,inf,sizeof(dp));

printf("Case #%d：%d\n",i,solve());

}

return 0;

}

**变式2：一个字符串有多少个回文子串**

#include <stdio.h>

#include <string.h>

#include <algorithm>

using namespace std;

const int mod = 10007;

char str[1005];

int dp[1005][1005];

int main()

{

int t,i,j,k,len,cas = 1;

scanf("%d",&t);

while(t--)

{

scanf("%s",str);

len = strlen(str);

for(i = 0; i<len; i++)

dp[i][i] = 1;//单个字符肯定是一个回文子串

for(i = 1; i<len; i++) //枚举终点

{

for(j = i-1; j>=0; j--) //枚举起点 ，这么枚举的原因是因为使字符串由小到大

{

//j~i的区间的回文数是j+1~i与j~i-1区间回文数的和，但是要注意这里会有重复的，运用的是容斥原理，注意此处算出的dp[i][j]只包含i或者j的，这里没有对同时包含i和j的情况的计算。非常重要的一点，也是难理解的一点

if(str[i] == str[j]) + dp[j][i] = (dp[j+1][i]+dp[j][i-1]-dp[j+1][i-1]mod)%mod;

//如果两边相等了，那么加上dp[j+1][i-1]的原因，是在两边同时加上相同的相同的字符串，加1的原因是因为加上1个相同的字符串"str[i]str[j]".

dp[j][i] = (dp[j][i]+dp[j+1][i-1]+1+mod)%mod;

//如果区间两头是相等的，则要加上dp[j+1][i-1]+1，因为首尾是可以组成一个回文子串的，而且首尾可以与中间任何一个回文子串组成新的回文子串

}

}

printf("Case %d: %d\n",cas++,dp[0][len-1]);

}

return 0;

}

**变式3：**

#include<iostream>//poj3280区间DP 最便宜的回文

using namespace std;

int w[30],n,m,dp[2005][2005];

char s[2005],ch;

int main()

{

int x,y;

while(cin>>n>>m)

{

cin>>s;

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

{

cin>>ch>>x>>y;

**w[ch-'a']=min(x,y);**

}

for(int i=m-1;i>=0;i--)

{

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

{

if(s[i]==s[j])

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

else

**dp[i][j]=min(dp[i+1][j]+w[s[i]-'a'],dp[i][j-1]+w[s[j]-'a']);**

}

}

cout<<dp[0][m-1]<<endl;

}

return 0;

}

**变式4：**

#include<bits/stdc++.h>//hdu2476 区间DP

using namespace std;

string s1,s2;

int L;

int dp[110][110];//i到j的刷法

int ans[110];

int solve()

{

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

{

for(int i=j;i>=0;i--)//i为头，j为尾

{

**dp[i][j]=dp[i+1][j]+1;//每个单独刷**

for(int k=i+1;k<=j;k++) //枚举分割点

{

if(s2[i]==s2[k])//i与k相同

{

**dp[i][j]=min(dp[i][j],dp[i+1][k]+dp[k+1][j]);//端点相等的情况下假设去掉左端点**

}

}

}

}

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

ans[i]=dp[0][i];

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

{

if(s1[i]!=s2[i])

{

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

**ans[i]=min(ans[i],ans[i]+dp[j+1][i]);**

}

else

**ans[i]=ans[i-1];//如果相等，当前位置可以不刷**

}

return ans[L-1];

}

int main()

{

while(cin>>s1>>s2)

{

L=s1.size();

memset(dp,0,sizeof(dp));

cout<<solve()<<endl;

}

return 0;

}

**环形区间DP板子题：**

#include<bits/stdc++.h>//hdu3506环形区间DP 猴子派对

using namespace std;

int n;

int a[2010];

int dp[2010][2010];

int sum[2010];

int solve()

{

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

{

**for(int i=1;i+len-1<=2\*n;i++)**

{

int j=i+len-1;

if(len==1) dp[i][j]=0;

else

{

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

{

int tmp=dp[i][k]+dp[k+1][j]+(sum[j]-sum[i-1]);

dp[i][j]=min(dp[i][j],tmp);

}

}

}

}

return dp[1][n];

}

int main()

{

while(cin>>n)

{

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

{

cin>>a[i];

**a[i+n]=a[i];//复制数组补长**

}

**for(int i=1;i<=2\*n;i++)**

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

memset(dp,0x7f7f7f,sizeof(dp));

cout<<solve()<<endl;

}

return 0;

}

**变式1：乘法型**

#include<bits/stdc++.h>//特殊环形区间DP ACwing 能量项链

using namespace std;

int n;

int a[210];

int dp[210][210];

int solve()

{

for(int len=3;len<=n+1;len++)//注意是n+1

{

for(int i=1;i+len-1<=2\*n;i++)

{

int j=len+i-1;

for(int k=i+1;k<j;k++)//注意是<

{

**int tmp=dp[i][k]+dp[k][j]+a[i]\*a[k]\*a[j];//注意两个都是k这里不能分割，和石子合并不同**

dp[i][j]=max(dp[i][j],tmp);

}

}

}

int maxv=-0x7f7f7f;

**for(int i=1;i<=n;i++)//重要**

**maxv=max(maxv,dp[i][n+i]);**

**return maxv;**

}

int main()

{

cin>>n;

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

{

cin>>a[i];

a[i+n]=a[i];

}

memset(dp,0,sizeof(dp));

cout<<solve()<<endl;

return 0;

}

**最大上升子序列：**

#include<iostream>

#include<algorithm>

using namespace std;

int a[1010],maxlen[1010];

int main()

{

int n;

cin>>n;

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

{

cin>>a[i];

maxlen[i]=1;

}

for(int i=1;i<=n;i++)//求以第i个数为终点的最长上升子序列的长度

{

for(int j=1;j<i;j++)//查看以第j个数为终点的最长上升子序列的值

{

if(a[i]>a[j])

{

**maxlen[i]=max(maxlen[i],maxlen[j]+1);**

}

}

}

**cout<<\*max\_element(maxlen+1,maxlen+1+n);//取最大**

return 0;

}

**最长公共子序列：**

#include<iostream>

#include<algorithm>

#include<cstring>

using namespace std;

char s1[1000],s2[1000];

int maxlen[1000][1000];

int main()

{

while(cin>>s1>>s2)

{

int l1=strlen(s1);

int l2=strlen(s2);

int ntmp;

int i,j;

for(i=0;i<=l1;i++)

maxlen[i][0]=0;

for(j=0;j<=l2;j++)

maxlen[0][j]=0;

for(int i=1;i<=l1;i++)

{

for(int j=1;j<=l2;j++)

{

**if(s1[i-1]==s2[j-1])**

**maxlen[i][j]=maxlen[i-1][j-1]+1;**

**else**

**maxlen[i][j]=max(maxlen[i-1][j],maxlen[i][j-1]);**

}

}

cout<<maxlen[l1][l2]<<endl;

}

return 0;

}

**最大子列和：**

#include<bits/stdc++.h>//hdu1003 最大子列和

using namespace std;

int n,t;

int a[100010];

int L,R,maxn;

int ans[25][4];

int dp[100010];//前i个数的最大和子列

void solve()

{

dp[1]=a[1];

L=1,R=1,maxn=dp[1];

int start=1;

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

{

**if(dp[i-1]+a[i]>=a[i]){**

**//即dp[i-1](以a[i-1]为结尾的子序列的和的最大值)的值为正，那么dp[i-1]则对dp[i]有贡献**

**dp[i]=dp[i-1]+a[i];**

}

else{

dp[i]=a[i];

start=i;//抛弃dp[i-1]，起点发生变化

}

if(dp[i]>maxn){

maxn=dp[i];

L=start;R=i;

}

}

}

int main()

{

cin>>t;

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

{

cin>>n;

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

{

cin>>a[j];

}

solve();

ans[i][1]=maxn;ans[i][2]=L;ans[i][3]=R;

}

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

{

cout<<"Case "<<i<<":"<<endl;

cout<<ans[i][1]<<" "<<ans[i][2]<<" "<<ans[i][3]<<endl;

}

return 0;

}