首先bitset优化要满足几个条件：

1.这个dp方程里面的数字只有可能是0或者1

2.dp方程的时间复杂度可能刚刚好超过时限，比方说1000的三次方，200的四次方，10000的平方,最后刚刚好可以通过/32来达到时限要求

然后就可以使用bitset进行优化，下面是例题：

BZOJ3687: 简单题

求一个集合的所有子集的算术和的异或和。a[i]>0，1<n<1000，∑ai≤2000000。

用f[i]表示i这个数会被拼出多少次，显然只有拼出奇数次才会有影响，故只用记录奇偶即可

而每次输入一个数x，就把f数组左移x位和原来的异或一下就显然是新的答案了

最后枚举一下哪些数可以即可，算法复杂度为O（n\*2000000/32）

bitset<2000000>f;

ll x,n,sum=0,ans=0;

int main()

{ll i;

cin>>n;

f[0]=1;

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

{

scanf("%lld",&x);

sum+=x;

f^=(f<<x);

}

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

if(f[i])ans^=i;

printf("%lld\n",ans);

return 0;

}

例题：BZOJ4484: [Jsoi2015]最小表示

对于一个N个点（每个点从1到N编号），M条边的有向图，JYY发现，

如果从图中删去一些边，那么原图的连通性会发生改变；而也有一些边，删去之后图的连通性并不会发生改变。

JYY想知道，如果想要使得原图任意两点的连通性保持不变，我们最多能删掉多少条边呢？

为了简化一下大家的工作量，这次JYY保证他给定的有向图一定是一个有向无环图

首先，一条边(u,v)可以删除的条件为：删除这条边后，仍然能从u走到v

这样的话我们可以贪心处理，对于两个点(u,v)，我们保留其最长的路径，其余的全部删去

具体实现的时候我们可以先来一边拓扑排序，同时记录下每个点出现的时间，以及该时间入队的点

一个点连出去最长的边一定是包含先访问的点(又是一个贪心)

然而正序处理的话我们并不知道一个点连到的点的联通性

因此我们倒序处理，

联通性用bitset维护

int VisitTime[MAXN];//i是第几个入队的

int InputTime[MAXN];//第i个入队的是谁

int inder[MAXN];

bitset<MAXN>can[MAXN];//联通性

int N,M;

int to[MAXN];

int comp(const int &a,const int &b)

{

return VisitTime[a]<VisitTime[b];

}

void Topsort()

{

queue<int>q;

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

if(inder[i]==0) q.push(i);

int tot=0;

while(q.size()!=0)

{

int p=q.front();q.pop();

InputTime[++tot]=p;

VisitTime[p]=tot;

for(int i=head[p];i!=-1;i=edge[i].nxt)

{

inder[edge[i].v]--;

if(inder[edge[i].v]==0)

q.push(edge[i].v);

}

}

int ans=0;

for(int i=N;i>=1;i--)

{

int x=InputTime[i],tot=0;

can[x][x]=1;

for(int j=head[x];j!=-1;j=edge[j].nxt)

to[++tot]=edge[j].v;

sort(to+1,to+tot+1,comp);

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

{

if(can[x][to[j]]) ans++;

else can[x]|=can[to[j]];

}

}

printf("%d",ans);

}