这里的题目方法不一定唯一

1.反悔贪心

如果当前解比原来的最优解更优，那就反悔更新全局最优解。用优先队列维护。

1.1 反悔堆

1.1.1 用时一定模型

有n项任务，每项任务都要花1单位时间，且有一个截至时间d_i,完成获得利润p_i,求最大利润

将任务按截止时间从小到大排序，还有时间做任务则做任务，否则若任务p_i大于当前已做任务的最小利润值，替换他。

1.1.2 价值一定模型

有n项任务，每项任务都有一个截至时间d_i和完成耗时t_i,求最多可完成任务数

将任务按截止时间从小到大排序，还有时间做任务则做任务，否则若任务t_i小于当前已做任务的最大利润值，替换他。

1.2 反悔自动机

1.2.1 堆反悔自动机

已知接下来n天的股票价格p_i，每天可以买入一股当天的股票，卖出一股已有的股票，或者什么都不做，求n天之后最大的利润。

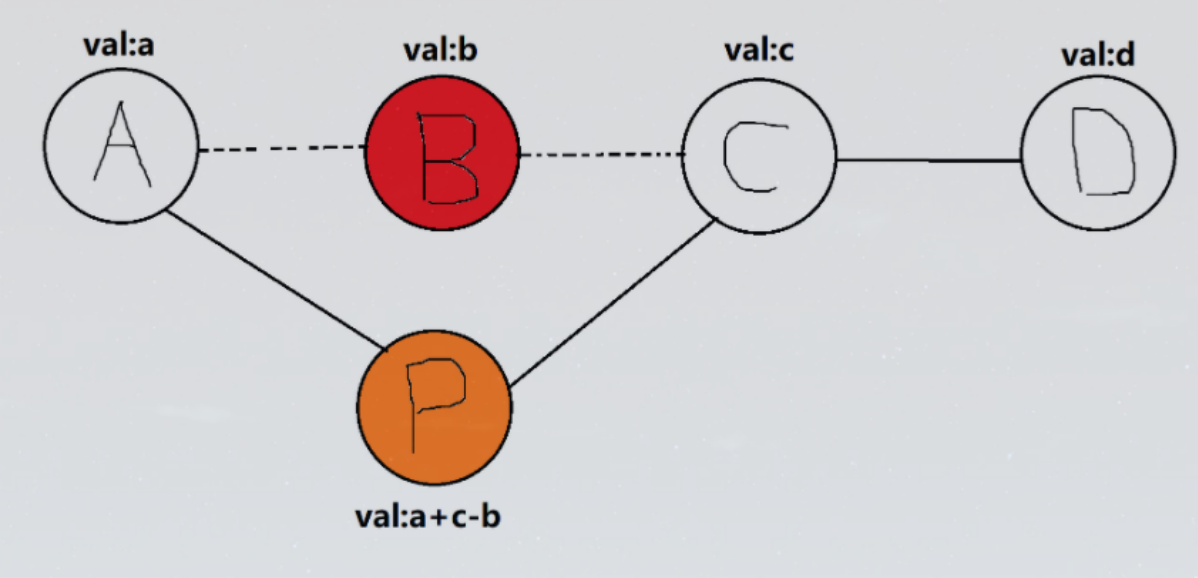
$$C_{cell}-C_{buy}=(C_{cell}-C_i)+(C_i-C_{buy})$$

上面的等式即被称为反悔自动机的反悔策略，因为我们并没有反复更新全局最优解，而是通过差值消去中间项的方法快速得到的全局最优解。将之前的p_i进行更新到优先队列中。

```
#include<bits/stdc++.h>
using namespace std;
int n,k;
priority_queue<int, vector<int>, greater<int> > q;
signed main() {
    cin>>n;
    int ans=0;
    for (int i = 1; i <= n; ++i) {
        cin>>k;//p_i
        if (!q.empty()&&q.top()<k)ans+=k-q.top(),q.pop(),q.push(k);
        q.push(k);
    }
    cout<<ans<<endl;
    system("pause");
    return 0;
}
```

1.2.2 双向链表反悔自动机

有n个位置，每个位置有一个价值。有m个树苗，将这些树苗种在这些位置上，相邻位置不能都种。求可以得到的最大值或无解信息。



选择B，新添一个点P，下次选到P代表反悔，会重新变回B，同时左右的点不能选择了

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
const int N=5e5+5;
struct node{
    int v;int id;
    bool operator <(const node &a)const {return v<a.v;}
};
priority_queue<node>q;
int a[N];
int l[N],r[N];
bool vis[N];
int n,m;
void del(int x){
    vis[l[x]]=vis[r[x]]=1;
    l[x]=l[l[x]];r[l[x]]=x;
    r[x]=r[r[x]];l[r[x]]=x;
}
signed main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        scanf("%lld",&a[i]);
        l[i]=i-1;r[i]=i+1;
        q.push({a[i],i});
    }
    r[0]=1;l[n+1]=n;
    int ans=0;
    for (int p=1;p<=m;p++){
        while(vis[q.top().id])q.pop();
        node t=q.top();q.pop();
        if(t.v<0)break;
        ans+=t.v;
        int x=t.id;
        a[x]=a[l[x]]+a[r[x]]-a[x];
        del(x);
        q.push({a[x],x});
    }
```

```
    cout<<ans<<endl;
    return 0;
}
```

长度为n的字符串s修改k个字符s能写出多少个子串ac，输出修改后的串

```
#include<bits/stdc++.h>
#define pii pair<int,int>
using namespace std;
const int M=5e5+9;
int n,k,ans;
int pre[M],nex[M],a[M],l[M],r[M];
char s[M];
bool vis[M];
priority_queue<pii>q;
int main(){
    scanf("%d%d%s",&n,&k,s+1);
    for(int i=1;i<=n;++i)pre[i]=i-1,nex[i]=i+1,l[i]=r[i]=i;
    for(int i=1;i<n;++i){
        a[i]=(s[i]!='a')+(s[i+1]!='c');
        q.push({-a[i],i});
    }
    a[0]=a[n]=1e8;
    nex[0]=1;pre[n+1]=n;
    while(!q.empty()&&-q.top().first<=k){
        int val=q.top().first,id=q.top().second;
        q.pop();
        if(vis[id])continue;
        k+=val;
        ans++;
        r[id]=r[nex[id]];
        l[id]=l[pre[id]];
        a[id]=a[pre[id]]+a[nex[id]]+val;
        vis[pre[id]]=vis[nex[id]]=1;
        pre[id]=pre[pre[id]];
        nex[id]=nex[nex[id]];
        pre[nex[id]]=nex[pre[id]]=id;
        q.push({-a[id],id});
    }
    printf("%d\n",ans);
    for(int id=1;id<=n;++id){
        if(vis[id])continue;
        for(int i=l[id]+1;i<r[id];i+=2)s[i]='a',s[i+1]='c';
    }
    printf("%s\n",s+1);
    return 0;
}
```