前置知识

四边形不等式

定义

$$\begin{gathered}\text{若函数}w(x,y)(\mathbb{Z}\times\mathbb{Z}\to\mathbb{Z})\text{对于}\forall a,b,c,d\in\mathbb{Z},\\ \text{其中}a\le b\le c\le d,\\ \text{都有}w(a,d)+w(b,c)\ge w(a,c)+w(b,d),\\ \text{则称函数}w\text{满足四边形不等式}\end{gathered}$$

推论

$$\begin{gathered}\text{若函数}w(x,y)(\mathbb{Z}\times\mathbb{Z}\to\mathbb{Z})\text{对于}\forall a,b\in\mathbb{Z},\text{其中}a<b,\\ \text{都有}w(a,b+1)+w(a+1,b)\ge w(a,b)+w(a+1,b+1),\\ \text{则称函数}w\text{满足四边形不等式}\end{gathered}$$

如果函数w满足四边形不等式，我们也称它满足凸完全单调性，或者说它是凸函数。

决策单调性

定义

$$\begin{gathered}\text{对于形如}f[i]=min_{0\le j<i}(f[j]+w(j,i))\text{的状态转移方程,}\\ \text{记}p[i]\text{为}f[i]\text{取到最小值时}j\text{的值,}p[i]\text{即为}f[i]\text{的最优决策。}\\ \text{如果}p[i]\text{在}[1,n]\text{上单调不减,则称}f\text{具有决策单调性。}\end{gathered}$$

也可理解为若w满足四边形不等式，则f具有决策单调性。

步骤

（1）更新，找到第一个能够涵盖i位置的决策点。

（2）通过二分找决策点。

（3）找到一个并没有被完全覆盖的队尾。

斜率优化

斜率优化是决策单调性的一种衍生算法，利用斜率使得决策优化能在线性时间内解决。

$$\text{将决策}j\text{看作平面上的一个点}(x_j,y_j),\text{对于}i\text{来说，}\forall k<j,f_j+w(j,i)\le f_k+f(k,i),\text{当且仅当}\frac{y_j-y_k}{x_j-x_k}\le C_i$$

$$\text{其中}x_j,y_j\text{均在计算}f_j\text{后已知，}C_i\text{已知，实现这一算法的一般方法为维护所有决策点构成的凸包：}$$

$$\begin{cases}1.\text{当}x,C\text{单调递增时可以利用单调队列直接维护凸包，询问时只需弹出队首不满足条件的元素即可。时间复杂度}O(n)\\2.\text{当}x\text{单调递增时，在凸包上二分寻找寻找第一个斜率不超过}C_i\text{的位置，时间复杂度}O(nlogn)\\3.\text{当}x,C\text{均不单调递增时可以使用}CDQ\text{分治，时间复杂度}O(nlogn)\end{cases}$$

注意事项

（1）写出dp转移方程后，先判断是否可以斜率优化（一般存在f_i*f_j项或者$\frac{Y(j)-Y(j')}{X(j)-X(j')}$）的形式）

（2）斜率的计算会因为下取整出现误差，因此用slope函数要返回long double类型。

（3）在比较两个斜率时，尽量写上等于，对于去重有奇效。（有重点时会导致斜率分母出锅）

（4）斜率单调暴力移指针，斜率不单调二分找答案。

例题

[NOI2009] 诗人小G

给出n个字符串，把这些字符串依次用空格（算一个长度）连接分成若干段，若一段长度为xx，那么代价是$|x-L|^P$,求代价和最小的方案，如果代价大于$1e18$

$1\le n\le 10^5,1\le L\le 3\times 10^6,1\le P\le 10$

```
#include<bits/stdc++.h>
#define inf 1e18
#define int long long
using namespace std;
const int maxn=5e5+7;
const int mod=1e9+7;

int t,n,L,P;
int sum[maxn],res[maxn];
int q[maxn],stk[maxn]; //找到1~n最优决策的每一段
char str[maxn][35];
long double dp[maxn];

long double fpow(long double a,int b){
    long double res=1;
    while(b){
```

```
        if(b&1) res=res*a;
        a=a*a;
        b>>=1;
    }
    return res;
}

inline long double calc(int j,int i){
    return dp[j]+fpow(abs(sum[i]-sum[j]+(i-j-1)-L),P);
}

void init(){
    for(int i=1;i<=n;i++){
        dp[i]=inf;
        res[i]=0;
    }
}

inline int bs(int a,int b){ //bin_search二分找到第一个决策b比决策a更优的位置
    if(calc(a,n)<calc(b,n)) return n+1;
    int l=b,r=n,mid;
    int ans=-1;
    while(l<=r){
        mid=(l+r)>>1;
        if(calc(b,mid)<=calc(a,mid)){
            ans=mid;
            r=mid-1;
        }else l=mid+1;
    }
    return ans;
}

signed main(){
    cin>>t;
    while(t--){
        cin>>n>>L>>P;
        init();
        for(int i=1;i<=n;i++){
            cin>>str[i];
            sum[i]=strlen(str[i])+sum[i-1];
        }
        int head=1,tail=0;
        q[++tail]=0;
        dp[0]=0;
        for(int i=1;i<=n;i++){
            while(head<tail&&bs(q[head],q[head+1])<=i) head++; //使得head决策点的对应区域
包含i

            res[i]=q[head];
            dp[i]=calc(q[head],i);
            while(head<tail&&bs(q[tail-1],q[tail])>=bs(q[tail],i))tail--;
            //把以队尾决策点为决策点不如以i为决策点更优的位置出队
            q[++tail]=i; //并替换成i
        }
        if(dp[n]>inf){
            printf("Too hard to arrange\n");
        }else{
            printf("%lld\n",(long long)dp[n]);
            int top=0;
            for(int i=n;i;i=res[i]) stk[++top]=i;
            stk[++top]=0;
```

```
            for(int i=top-1;i>=1;i--){
                int r=stk[i],l=stk[i+1]+1;
                for(int j=l;j<r;j++) printf("%s ",str[j]);
                printf("%s\n",str[r]);
            }
        }
        printf("--------------------\n");
    }
    return 0;
}
```

[HNOI2008]玩具装箱(斜率优化入门题)

有 n个玩具，第 i个玩具的价值为 c_i 。要求将这 n个玩具排成一排，分成若干段。对于一段 $[l,r]$，它的代价为 $(r-l+\sum_{i=l}^{r}c_i-L)^2$，其中 L 为一个常量，求分段的最小代价。

$1\le n\le 5\times 10^4, 1\le L\le 10^7, 1\le C_i\le 10^7$

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
#define int long long
#define ld long double
using namespace std;
const int maxn=5e4+7;

int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return x*f;}

int n,L,head=1,tail=0,tmp;
int Q[maxn],S[maxn],dp[maxn];

inline int X(int x){return S[x];}

inline int Y(int x){return dp[x]+(S[x]+L)*(S[x]+L);}

inline ld slope(int a,int b){
    return (ld)(Y(a)-Y(b))/(X(a)-X(b));
}

signed main(){
    n=read();L=read();
    ++L;
    memset(S,0,sizeof(S));
    for(int i=1;i<=n;i++) S[i]=read()+S[i-1]+1;
    Q[++tail]=0;
    for(int i=1;i<=n;i++){
        while(head<tail&&slope(Q[head],Q[head+1])<=2*S[i]) ++head;
        tmp=Q[head]; //决策点
        dp[i]=dp[tmp]+(S[i]-S[tmp]-L)*(S[i]-S[tmp]-L);
        while(head<tail&&slope(Q[tail-1],Q[tail])>=slope(Q[tail-1],i)) --tail;
        Q[++tail]=i;
    }
    printf("%lld",dp[n]);
    return 0;
}
```

[SDOI2012]任务安排

n个任务，每个任务有一个完成所需时间t和一个代价c，可以将它们任意分成几份，但不能改变其顺序。开始每组任务需要一个s的时间。每组任务的代价为该组任务的代价和乘以该组任务的完成时刻。

```
#include<bits/stdc++.h>
#define inf 0x3f3f3f3f
#define int long long
#define db long double
using namespace std;
const int maxn=300007;
const int mod=1e9+7;

int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return x*f;}

int n,S,t[maxn],c[maxn];
int q[maxn],dp[maxn];
int L,R;

int solve(int val){
    int l=L,r=R-1,ans=-1;
    while(l<=r){
        int m=(l+r)>>1;
        if((dp[q[m+1]]-c[q[m+1]]*S-dp[q[m]]+c[q[m]]*S)>val*(c[q[m+1]]-
c[q[m]]))ans=m,r=m-1;
        else l=m+1;
    }
    if(ans==-1)return q[R];
    return q[ans];
}

signed main(){
    n=read();
    S=read();
    memset(dp,inf,sizeof(dp));
    dp[0]=0;q[0]=0;
    for(int i=1;i<=n;i++){
        t[i]=read()+t[i-1];
        c[i]=read()+c[i-1];
    }
    L=0,R=0;
    for(int i=1;i<=n;i++){
        int j=solve((db)t[i]);
        dp[i]=dp[j]+t[i]*(c[i]-c[j])+S*(c[n]-c[j]);
        while(L<R&&(dp[q[R]]-c[q[R]]*S-dp[q[R-1]]+c[q[R-1]]*S)*(c[i]-c[q[R]])>=
(dp[i]-c[i]*S-dp[q[R]]+c[q[R]]*S)*(c[q[R]]-c[q[R-1]])) R--;
        q[++R]=i;
    }
    cout<<dp[n]<<"\n";
    return 0;
}
```

[CEOI2004]锯木厂选址

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#define db double
using namespace std;
const int M=3e4+1;
int n;
int q[M],fi,la,ans=2e9+1;
int sum,s[M],d[M],w[M];
db calc(int j,int k){return 1.0*(d[j]*s[j]-d[k]*s[k])/(s[j]-s[k]);}
int count(int i,int j){return sum-d[j]*s[j]-d[i]*(s[i]-s[j]);}
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++){scanf("%d%d",&w[i],&d[i]);}
    for(int i=n;i>=1;i--) d[i]+=d[i+1];
    for(int i=1;i<=n;i++) s[i]=s[i-1]+w[i],sum+=d[i]*w[i];
    for(int i=1;i<=n;i++){
        while(fi<la&&calc(q[fi],q[fi+1])>d[i]) ++fi;
        ans=min(ans,count(i,q[fi]));
        while(fi<la&&calc(q[la-1],q[la])<calc(q[la],i)) --la;
        q[++la]=i;
    }
    printf("%d\n",ans);
    return 0;
}
```

P2254 [NOI2005] 瑰丽华尔兹

$n \times m$ 的矩阵。以 (x, y) 为起点。一共 k段时间，每段时间为 $[s_i, t_i](t_i + 1 = s_{i+1})$，每秒可以向 d_i方向运动一个单位（**不能超出矩阵，不能走到给出矩阵的障碍物处**，$d = \{1, 2, 3, 4\}$分别表示上下左右）或不动，求最后运动最长总距离。

```
/*首先考虑对于时间t来dp:
f[t][i][j]表示在第t时刻在第i行第j列所能获得的最长距离。
转移方程：f[t][i][j]=max(f[t-1][i][j],f[t][i*][j*]+1)(i*,j*为上一个合理的位置)
这样时间复杂度为O(TNM)，可以过50%,但对于100%TLE且MLE。
所以必须优化，首先把时间t换成区间k,
令f[k][i][j]表示在第k段滑行区间中在位置i，j所能获得最长距离
注意到在第k段时间内只能向某个方向最多走x步（x为区间长度），得到转移方程
f[k][i][j]=max(f[k-1][i][j],f[k][i*][j*]+dis(i,j,i*,j*))(i*,j*为上一个合理的位置)
这个做法的时间复杂度是O(kn^3)，会超时，需要进一步优化
用单调队列优化掉内层的一个n，就可以做到O(kn^2)，可以AC，本代码中还使用了滚动数组优化
用单调递减队列求最大值时，遇到障碍清空整个队列即可，另外队列比较时需要加上偏移量dis*/
#include<bits/stdc++.h>
#define mk make_pair
#define pii pair<int,int>
#define F first
#define S second
using namespace std;
const int N=205;

int n,m,sx,sy,k,ans,dp[N][N];
char mp[N][N];
int dx[]={0,-1,1,0,0},dy[]={0,0,0,-1,1};
pii q[N];
```

```cpp
void work(int x,int y,int len,int d){
    int head=1,tail=0;
    for(int i=1;x>=1&&x<=n&&y>=1&&y<=m;i++,x+=dx[d],y+=dy[d]){
        if(mp[x][y]=='x') head=1,tail=0;
        else{
            while(head<=tail&&q[tail].S+i-q[tail].F<dp[x][y]) tail--;
            q[++tail]=mk(i,dp[x][y]);
            if(q[tail].F-q[head].F>len) ++head;
            dp[x][y]=q[head].S+i-q[head].F;
            ans=max(ans,dp[x][y]);
        }
    }
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    cin>>n>>m>>sx>>sy>>k;
    for(int i=1;i<=n+1;++i) mp[i][0]=mp[i][m+1]='x';
    for(int i=1;i<=m+1;++i) mp[0][i]=mp[n+1][i]='x';
    memset(dp,0xf3,sizeof(dp));
    dp[sx][sy]=0;
    for(int i=1;i<=n;++i){
        for(int j=1;j<=m;++j){
            cin>>mp[i][j];
        }
    }
    for(int i=1,s,t,d,len;i<=k;++i){
        cin>>s>>t>>d;
        len=t-s+1;
        if(d==1) for(int i=1;i<=m;++i) work(n,i,len,d);
        if(d==2) for(int i=1;i<=m;++i) work(1,i,len,d);
        if(d==3) for(int i=1;i<=n;++i) work(i,m,len,d);
        if(d==4) for(int i=1;i<=n;++i) work(i,1,len,d);
    }
    cout<<ans<<"\n";
    return 0;
}
```

参考资料

https://www.cnblogs.com/birchtree/p/12937975.html

https://www.cnblogs.com/Xing-Ling/p/11210179.html

https://alan-sp.github.io/post/jue-ce-dan-diao-xing-xue-xi-bi-ji/

https://www.cnblogs.com/kebingyi/p/14157680.html