# 李超树

---

## 解决问题

n条线段，问你当 $x = x_0$ 时，最大的 y 是多少

---

## 思路

将直线一条一条地插入到线段树，当这条直线要更新某个区间的时。

1.**区间长度为1

1.~~新线段斜率>旧线段斜率~~。 若对于当前区间中点，新线段y值大于旧线段，更新T，递归检查左子区间。 否则，递归检查右子区间。

2.~~新线段斜率≤旧线段斜率~~。若对于当前区间中点，新线段y值大于旧线段，更新T，递归检查右子区间。 否则，递归检查左子区间。

---

## 模板

### 代码

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn = 1e6+10;、

/*直线斜率，截距*/
ll k[maxn], b[maxn];
/*函数值计算*/
ll f(int cur, int x){return k[cur]*x + b[cur];}
/*线段树，存直线编号*/
int T[maxn*4];

void build(int p,int l,int r,int cur){
    T[p] = cur;
    if(l == r) return;
    int mid = (l+r)/2;
    build(p*2,l,mid,cur),
```

```
    build(p*2+1,mid+1,r,cur);
}

void update(int p,int l,int r,int cur){
    int pre = T[p];
    if(l == r){
        if(f(pre, l) < f(cur, l)) T[p] = cur;
        return;
    }
    int mid = (l+r)/2;
    /*新线段斜率高于原线段*/
    if(k[cur] > k[pre]){
        if(f(cur, mid) > f(pre, mid))
            T[p] = cur, update(p*2,l,mid,pre);
        else
            update(p*2+1,mid+1,r,cur);
    }
    /*新线段斜率低于原线段*/
    else{
        if(f(cur, mid) > f(pre, mid))
            T[p] = cur, update(p*2+1,mid+1,r,pre);
        else
            update(p*2,l,mid,cur);
    }
}

ll query(int p, int l, int r, int x)
{
    ll ans = f(T[p], x);
    if(l == r) return ans;
    int mid =(l+r)/2;
    if(x <= mid) ans = max(ans, query(p*2,l,mid,x));
    else ans = max(ans, query(p*2+1,mid+1,r,x));
    return ans;
}
```

---

# 例题： Washing clothes

## 题意

有n个人，每个人有一件衣服需要洗，可以自己手洗花费t时间，也可以用洗衣机洗，洗衣机只有一台，现在给你每个人最早可以开始洗衣服的时间，问当洗衣机的洗衣时间分别为1, 2....t的时候洗完所有衣服的最短时间。

## 思路

1.首先把按照洗衣服时间排序，最终洗衣服时间的瓶颈在于最后的若干人。问题转化为 **对于当前的x，确定哪些人用洗衣机。**

2.用g(i)表示从第i个人开始后面全用洗衣机时，手洗的结束时间。f(i)表示从第i个人开始后面全用洗衣机时，机洗的结束时间。那么:

> $g(i) = t_{i-1} + y$
>
> $f(i) = max(t_j + (n - j + 1)x)_{(j \geq i)}$

从i开始的人用洗衣机的时间是 **max( g(i), f(i))** 。

3. g(i)随着i递增，f(i)随着i递减。答案是一个凸函数，有极小值点。而f(i)函数，是随着x的增加而递增的，所以x增加时极小值点会右移，我们枚举x，维护这个极小值点即可。x确定时，用李超线段树可以在O(log(n))的时间内快速得到f(i)。

## 代码

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn = 1e6+10;
int N, y;
ll t[maxn], k[maxn], b[maxn], ans[maxn];
int T[maxn*4];
ll f(int cur, int x){return k[cur]*x + b[cur];}//函数值计算
void build(int p,int l,int r,int cur){
    T[p] = cur;
    if(l == r) return;
    int mid = (l+r)/2;
    build(p*2,l,mid,cur),
    build(p*2+1,mid+1,r,cur);
}
void update(int p,int l,int r,int cur){
    int pre = T[p];
    if(l == r){
        if(f(pre, l) < f(cur, l)) T[p] = cur;
        return;
    }
    int mid = (l+r)/2;
    if(k[cur] > k[pre]){
        if(f(cur, mid) > f(pre, mid))
            T[p] = cur, update(p*2,l,mid,pre);
        else
            update(p*2+1,mid+1,r,cur);
    }
    else{
        if(f(cur, mid) > f(pre, mid))
```

```
            T[p] = cur, update(p*2+1,mid+1,r,pre);
        else
            update(p*2,l,mid,cur);
    }
}
ll query(int p, int l, int r, int x)
{
    ll ans = f(T[p], x);
    if(l == r) return ans;
    int mid =(l+r)/2;
    if(x <= mid) ans = max(ans, query(p*2,l,mid,x));
    else ans = max(ans, query(p*2+1,mid+1,r,x));
    return ans;
}
void solve(){
    build(1, 0, y, N);//一开始存在线段N
    int p = N-1;
    for(int x = y; x > 0; --x){
        ll cur = max(t[p]+y, query(1, 0, y, x));
        ll res = query(1, 0, y, x);//所有加入的线段中f(x)最大值
        while(p > 0 && max(t[p-1]+y, max(res,f(p, x))) <= cur){
            update(1, 0, y, p); //加入线段
            res = max(res, f(p, x));
            cur = max(t[p-1]+y, res);//更新当前值
            p--;
        }
        ans[x] = cur;
    }
    for(int i = 1; i <= y; ++i){
        printf("%lld%c", ans[i],(i==y)?'\n':' ');
    }
}
int main(){
    while(scanf("%d%d",&N,&y) == 2){
        t[0] = -y;
        for(int i = 1; i <= N; ++i) scanf("%lld", &t[i]);
        sort(t, t+N+1);
        for(int i = 0; i <= N; ++i){
            k[i] = N-i+1; b[i] = t[i];
        }
        solve();
    }
}
```