1. **页码:9　　行数:11　　印次:1**

**该行缩进2字符：**

(8) 若继续查询C 2，…

1. **页码:23　　行数:4　　印次:1**

**删除**“,int n”

void Swim(int k~~,int n~~) {

1. **页码:41　　行数:4　　印次:1**

**删除4-26行，即从题解开始到训练4之间的部分。更新为以下内容：**

**题解：**本题穷举超时，但后台数据较弱，二分也可通过。实际上不满足单调性，不可二分。例如，6 199 1 1 100 100 1 1，二分首先分3段，不行然后分4段，实际上分2段即可。本题可以采用RMQ+优化，当区间长度和上次相同时，不再重新计算，直接累加下一个区间最值即可。

1．算法设计

首先创建ST表，然后枚举分段1~*n*，用pre记录上次分段的区间长度，如果当前区间长度与上次相同时，直接累加下一区间最值。例如，*t*=10/4=2，分成4段，区间长度为2，累加区间[1,2]，[3,4]，[5,6]，[7,8]的最值，*t*=10/5=2，分成5段，区间长度和上次相同，直接累加[9,10]的最值，不必再重新累加5个分段的最值。

2．算法实现

void solve(){

ST(n);

int i,j=0,t,ans=0,pre=0;

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

t=n/i;

if(pre!=t){//如果区间长度和上次一样，不用初始化，直接累加下一个区间即可

j=0; ans=0;

}

for(;j<i;j++)

ans+=RMQ(t\*j+1,j\*t+t);

pre=t;

if(ans>k) break;

}

if(i>n) printf("-1\n");

else printf("%d\n",i);

}

1. **页码:63　　行数:-5　　印次:1**

**“ 颗 ”改为 “ 棵 ”：**

* sum[5]：左侧有一棵子树，其根为*c*[4]，sum[5]=*c*[4]+*c*[5]。

1. **页码:76　　行数:8　　印次:1**

**“ 5 ”改为 “ 4 ”：**

* Q 1：查询以1号节点为根的子树中的苹果数量，1号节点的进出序号为L[1]=1、R[1]=5，查询[1,5]的区间和，sum(R[1])−sum(L[1]−1)=4−0=4。所以1号节点的子树中的苹果数量为4。

1. **页码:193　　行数:25　　印次:1**

**添加 “ 子 ”：**

for(int j=1;j<=top;j++)//查询完毕后才把这些积映射到节点

query(dep[j],id[j]);//在第1个棵子树中查询时，只有树根有映射

1. **页码:201　　行数:5　　印次:1**

**删除 “ *n* ”：**

都减少一半，可以保证时间复杂度为*O*(*~~n~~*log*n*)。菊花图转变为二叉树后如下图所示。

1. **页码:204　　行数:7　　印次:1**

**删除 “ z ”：**

C 1~~z~~

1. **页码:206　　行数:5　　印次:1**

**修改本页两个图中节点9的子树大小为 “ 5”，修改图中节点10的子树大小为 “ 2”：**





1. **页码:207　　行数:-1　　印次:1**

**修改图中*T*[2].ans=4为 “ *T*[2].ans=2 ”：**

*T*

[

1

].

rt

=

1

*T*

[

1

].

*q*

1 0

4 2

7 2

3 1

6 5

5 5

2 3

*u* dis

*T*

[

1

].

midlen

=

0

*T*

[

2

].

rt

=

9

*T*

[

3

].

rt

=

1

*T*

[

3

].

*q*

1 0

6 5

5 5

2 3

*T*

[

2

].

*q*

4 2

7 2

3 1

*T*

[

2

].

ans

=

4

*T*

[

3

].

ans

=

5

*T*

[

1

].

ans

=

7

1

2

3

=2

1. **页码:238　　行数:3　　印次:1**

**修改图中M为I的右孩子：**



1. **页码:239　　行数:3　　印次:1**

**修改图中M为I的右孩子（左、右图两处）：**



**修改图中M为I的右孩子（左图1处）：**



1. **页码:241　　行数:1　　印次:1**

**修改图中M为I的右孩子（左图1处）：**



1. **页码:242　　行数:1　　印次:1**

**修改图中M为I的右孩子（1处，与上面238页相同）：**



**修改图中M为I的右孩子（左图1处，与上面241页相同）：**



1. **页码:243　　行数:1　　印次:1**

**修改图中M为I的右孩子（1处，与上面238页相同）：**



**修改图中M为I的右孩子（左图1处）：**



1. **页码:324　　行数:-3　　印次:1**

**“右子节点”改为“右子树”：**

（1）右旋。*x*右旋时，携带自己的右子树向右旋转到*y*的右子树位置，

1. **页码:325　　行数:9　　印次:1**

**“右旋”改为“左旋”，“左子节点”改为“左子树”：**

（2）左旋。*x*左旋时，携带自己的左子树向左旋转到*y*的左子树位置，

1. **页码:444　　行数:25　　印次:1**

**“max”改为“min”：**

**状态转移方程：**dp[*j*]=min{dp[*j*],dp[*j*−*w*[*i*]]+val[*i*]}。

1. **页码:446　　行数:17　　印次:1**

**修改红色部分(4处)：**

**算法代码：**

void multi\_knapsack2(int n,int W){//二进制拆分

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

if(c[i]\*w[i]>=W){//转化为完全背包问题

for(j=w[i];j<=W;j++)

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

}

else{

for(int k=1;c[i]>0;k<<=1){//二进制拆分

int x=min(k,c[i]);

for(int j=W;j>=w[i]\*x;j--)//转化为01背包问题

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

c[i]-=x;

}

}

}

}

1. **页码:446　　行数:-4　　印次:1**

**删除原内容，添加红色新内容，修改如下：**

3．数组优化

若不要求最优性，仅关注可行性（如面值是否能拼成POJ1276），可使用数组优化。~~用num[~~*~~j~~*~~]数组记录容量为~~*~~j~~*~~时放入了多少个第~~*~~i~~*~~种物品，以实现物品数量限制。~~

**算法代码：**

~~void multi\_knapsack3(int n,int W){//数组优化~~

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

~~memset(num,0,sizeof(num));//统计数量~~

~~for(int j=w[i];j<=W;j++){~~

~~if(dp[j]<dp[j-w[i]]+v[i]&&num[j-w[i]]<c[i]){~~

~~dp[j]=dp[j-w[i]]+v[i];~~

~~num[j]=num[j-w[i]]+1;~~

~~}~~

~~}~~

~~}~~

~~}~~

bool dp[maxc];//dp[j]表示是否能够拼出金额j

int num[maxc];//num[j]表示金额为j时用了多少个第i种钱币

void multi\_knapsack3(int n,int W){//数组优化

ans=0,dp[0]=1;

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

memset(num,0,sizeof(num));//统计数量

for(int j=v[i];j<=W;j++){

if(!dp[j]&&dp[j-v[i]]&&num[j-v[i]]<c[i]){

dp[j]=1;

num[j]=num[j-v[i]]+1;

ans=max(ans,j);

}

}

}

}

1. **页码:450　　行数:11　　印次:1**

**添加一个右中括号：**



1. **页码:459　　行数:12　　印次:1**

**乱码行“459n tans=0;”改为“ int ans=0;”：**

int ans=0;

1. **页码:459　　行数:-3　　印次:1**

**修改红色部分：**

（5）若*a*[*i*]<*d*[len]，则将*a*[*i*]替换*d*[]中第1个大于*a*[*i*]的数。在*d*[]中查找第1个大于等于*a*[*i*]的数时，可以采用二分查找（*d*[]自身有序），也可以直接调用lower\_bound()函数，该函数也是采用二分查找实现的，每次查找的时间复杂度都为*O*(log*n*)。

1. **页码:460　　行数:2　　印次:1**

**修改红色部分，添加“等于”：**

*a*[*i*]<*d*[len]，则将*a*[*i*]替换*d*[]中第1个大于等于*a*[*i*]的数，

**倒数第5行代码，**upper\_bound()**改为**lower\_bound()**：**

\*lower\_bound(d+1,d+len+1,a[i])=a[i];

1. **页码:481　　行数:20　　印次:1**

**修改红色部分，添加“不”：**

* 若*u*是服务器，则每个子节点既可以是服务器，也可以不是服务器。
* 若*u*不是服务器，*u*的父节点是服务器，则每个子节点都不是服务器。
* 若*u*不是服务器，*u*的父节点也不是服务器，则*u*恰好有一个子节点是服务器。

1. **页码:500　　行数:21　　印次:1**

**修改红色部分，添加“**+dp[1][5]**”：**

（4）低位：在高位38已确定的情况下，统计首位是0～5的1位数，即380～385符合条件的数。最后1个数是386，没有计算，正好多计算了1个数000，不用特殊处理。ans+=dp[1][0]+dp[1][1]+dp[1][2]+dp[1][3]+dp[1][4]+dp[1][5]=307。

1. **页码:617　　行数:20　　印次:1**

**修改红色部分，添加“**+*n*×*m***”：**

（6）若从柱子id不可以直接跳出网格边界，则从id+*n*×*m*到与id的距离不大于*d*的网格id2引一条边(id+*n*×*m*, id2, inf)。注意：这里的距离是曼哈顿距离（行列差的绝对值之和）。

1. **页码:619　　行数:15　　印次:1**

**图中添加一条6—3的边，边权为10：**



1. **页码:625　　行数:9　　印次:1**

**输入样例少了一行，添加一行：1 2**

|  |  |
| --- | --- |
| 输入样例  5 5  2 2 5  3 2 3 4  2 1 5  3 1 2 5  1 2 | 输出样例  4 |

1. **页码:632　　行数:-7　　印次:1**

**A改为B：**

**题解：**本题要求最低总成本，包括一些模块在A上的运行成本、一些模块在B上的运行成本及在A、B两个模块中进行数据交换的成本。若将在A上运行的模块看作S集合，将在B上运行的模块看作T集合，则本问题转化为求最小割的问题。