2022 – 11 – 14模拟赛总结

好像开摆了？（~~策略重大失误，对自己实力认知不清~~）

题目	期望	实际	δ
$substr$	$0pts$	$10pts$	$+10pts$
$flower$	$40pts$	$20pts$	$-20pts$
$refract$	$0pts$	$0pts$	$0pts$
$paint$	$10pts$	$8pts$	$-2pts$

$T1$

考场：

字符串，不是我很擅长的类型，想了十多分钟没啥思路就跳了，只是在最后的时候随便乱写了一下，喜提 $10pts$，后面听说暴力其实有 $70pts$，有亿点点后悔。

改题：

多个字符串相互匹配使母串最短，其实要往建立 $trie$ 树这方面想，然后再利用 AC 自动机，建立 $fail$ 指针和字符图，在字符图上跑出代价最小可以包含所有串而且字典序最小的答案。（~~但是谁想得到啊~~，字符串太弱了，题做少了没什么感觉，或许要恶补一下了）。

具体的，我们在建立 $trie$ 树的时候，对一个字符串的末尾打上状态标记，利用状压的思想，在建立 $fail$ 指针和字符图的时候，从 $fail$ 指针那里继承状态（即子串的包含关系）。

然后我们从根节点开始跑 BFS，按照字典序扩展，和一般的 BFS 一样，所有扩展的代价都为 1，所以第一次扩展到所有状态的时候答案就是最优解，存答案可以模拟链表，对每个答案字符都存一个 pre 指针，最后递归找到答案，倒序输出即可。

CODE

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
template<typename T>inline void read(T &x){
    x=0;
    char c=getchar();
    T ret=0;
    while(!isdigit(c))ret|=!(c^'-'),c=getchar();
    while(isdigit(c))x=(x<<3)+(x<<1)+(c^48),c=getchar();
    if(ret)x=(~x)+1;
    return;
}
template<typename T>inline void print(T x){
    if(x<0)putchar('-'),x=(~x)+1;
    if(x>9)print(x/10);
    putchar((x-x/10*10)^48);
```

```
    return;
}
template<typename T>inline void wr1(T x){
    print(x);
    putchar(' ');
    return;
}
template<typename T>inline void wr2(T x){
    print(x);
    putchar('\n');
    return;
}
const ll N=610;
const ll M=1<<12|1;
int n,cnt,zt[N],trie[30][N],fail[N],ans[N*M],pre[N*M],no,tot,out[N],OUT;
bool vis[N][M];
char s[60];
queue<int>q;
queue<pair<int,int>>Q;
inline void build(){
    for(int i=0;i<26;++i){
        if(trie[i][0]){
            q.push(trie[i][0]);
        }
    }
    while(!q.empty()){
        int x=q.front();
        q.pop();
        for(int i=0;i<26;++i){
            if(trie[i][x]){
                fail[trie[i][x]]=trie[i][fail[x]];
                zt[trie[i][x]]|=zt[trie[i][fail[x]]];
                q.push(trie[i][x]);
            }
            else{
                trie[i][x]=trie[i][fail[x]];
            }
        }
    }
    return;
}
int main(){
    freopen("substr.in","r",stdin);
    freopen("substr.out","w",stdout);
    read(n);
    for(int i=1;i<=n;++i){
        memset(s,0,sizeof s);
        scanf("%s",s+1);
        int now=0;
        for(int j=1;j<=(int)strlen(s+1);++j){
            if(!trie[s[j]-'A'][now])trie[s[j]-'A'][now]=++cnt;
            now=trie[s[j]-'A'][now];
        }
        zt[now]|=1<<(i-1);
    }
```

```
    build();
    Q.push({0,0});
    vis[0][0]=1;
    while(!Q.empty()){
        int x=Q.front().first;
        int ZT=Q.front().second;
        Q.pop();
        if(ZT==((1<<n)-1)){
            while(no){
                out[++OUT]=ans[no];
                no=pre[no];
            }
            while(OUT)putchar(out[OUT]+'A'),OUT--;
            return 0;
        }
        for(int i=0;i<26;++i){
            if(!vis[trie[i][x]][ZT|zt[trie[i][x]]]){
                vis[trie[i][x]][ZT|zt[trie[i][x]]]=1;
                Q.push({trie[i][x],ZT|zt[trie[i][x]]});
                pre[++tot]=no;
                ans[tot]=i;
            }
        }
        ++no;
    }
    return 0;
}
```

$T2$

考场：

直接暴力枚举区间内所有的花求最优答案，$O(nm)$，能拿 $20pts$。

然后看到数据范围，还有部分分可以拿，对于 $a_I \leq 300$ 的数据点，可以维护 301 个线段树求解，$O(m \log n)$，也有 $20pts$。线段树的 $build$ 函数递归边界没 $return$ 我是没有想到的，~~这比错误好久没犯过子~~，看来后面还得把数据范围分治的程序分别运行一下。

改题：

奇奇怪怪的分块求解。分块的知识又增加了。

可以注意到，$k \leq 1e5$，那么其实是可以对每个块内每个可能的 k 预处理出块内最优答案的。

为了方便卡空间，我们固定块长为 1000。

我们首先对序列进行分块，然后分别对每个区间预处理。

具体的，我们把该区间内的值映射到一个值域空间上，我们通过刷表填充整个空间，每个下标表示小于等于该下标的最大值，根据定义，答案是模 k 意义下的最大值，那么每个 $[k \times i, k \times i + k - 1]$ 值域内的最大值模 k 之后取最大值就是该块模 k 意义下的最优答案。

然后查询就是类似普通的分块的查询。

复杂度最大在预处理，$O(\sqrt{n}k \ln k)$。

CODE

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
template<typename T>inline void read(T &x){
    x=0;
    char c=getchar();
    T ret=0;
    while(!isdigit(c))ret|=!(c^'-'),c=getchar();
    while(isdigit(c))x=(x<<3)+(x<<1)+(c^48),c=getchar();
    if(ret)x=(~x)+1;
    return;
}
template<typename T>inline void print(T x){
    if(x<0)putchar('-'),x=(~x)+1;
    if(x>9)print(x/10);
    putchar((x-x/10*10)^48);
    return;
}
template<typename T>inline void wr1(T x){
    print(x);
    putchar(' ');
    return;
}
template<typename T>inline void wr2(T x){
    print(x);
    putchar('\n');
    return;
}
const int cut=1000;
const int N=1e5;
int n,m,a[114514],mx[114514],ans[105][114514],t;
int main(){
    freopen("flower.in","r",stdin);
    freopen("flower.out","w",stdout);
    read(n);
    t=n/cut+1;
    read(m);
    for(int i=1;i<=n;++i){
        read(a[i]);
    }
    for(int i=0;i<t;++i){
        memset(mx,0,sizeof mx);
        for(int j=i*cut+1;j<=min(i*cut+cut,n);++j){
            mx[a[j]]=a[j];
        }
        for(int j=1;j<=N;++j){
            if(!mx[j])mx[j]=mx[j-1];
        }
        for(int j=1;j<=N;++j){
            for(int k=0;k<=N;k+=j){
                ans[i][j]=max(ans[i][j],mx[min(k+j-1,N)]%j);
            }
        }
```

```
    }
    while(m--){
        int x,y,k;
        read(x);
        read(y);
        read(k);
        int l=x/cut+1,r=y/cut;
        int out=0;
        if(l==r+1){
            for(int i=x;i<=y;++i){
                out=max(out,a[i]%k);
            }
        }
        else{
            for(int i=x;i<=l*cut;++i){
                out=max(out,a[i]%k);
            }
            for(int i=l;i<r;++i){
                out=max(out,ans[i][k]);
            }
            for(int i=r*cut+1;i<=y;++i){
                out=max(out,a[i]%k);
            }
        }
        wr2(out);
    }
    return 0;
}
```

$T3$

考场：

花时间最多的一道题，结果保龄了。。。

一开始根据条件考虑的是对 y 排序之后依次往下统计答案，可以利用树状数组求前缀和的同时储存，但发现好像会统计到错误的答案，于是就开始考虑把多余的答案减去，于是写了一个主席树和线段树去维护，但是调了很久一直是错的，无法通过大样例，于是先去做其他题换换思路，然后这题就寄了。

改题：

考虑 DP，既然通过 y 排序答案比较难统计，那么就对 x 排序来 DP。

定义 $DP[i][0]$ 表示以 i 为起点，往左连边的可行方案数，$DP[i][1]$ 则是往右边连边，显然初值都为 1（因为只走自己是一种方案，这里重复计算了只走自己的方案，最后统计答案的时候要减去）。

对于当前节点 i，考虑遍历之前所有节点，记遍历到的节点为 j，那么：

1. $y_j < y_i$ 时，$DP[i][0]+=DP[j][1]$，根据定义，这是合法的，因为从 j 往右连的节点绝对小于 i（比 i 大的根本就还没有被遍历过）。
2. $y_j > y_i$时，$DP[j][1]+=DP[i][0]$，根据定义，当前只有 （j,i） 之间的左连方案被统计到了 $DP[i][0]$，所以当前所有方案也适用于先从 j 往右连到 i。

最后统计每个点为起点的方案总数即可，复杂度 $O(n^2)$。

CODE

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const ll mod=1e9+7;
template<typename T>inline void read(T &x){
    x=0;
    char c=getchar();
    T ret=0;
    while(!isdigit(c))ret|=!(c^'-'),c=getchar();
    while(isdigit(c))x=(x<<3)+(x<<1)+(c^48),c=getchar();
    if(ret)x=(~x)+1;
    return;
}
template<typename T>inline void print(T x){
    if(x<0)putchar('-'),x=(~x)+1;
    if(x>9)print(x/10);
    putchar((x-x/10*10)^48);
    return;
}
template<typename T>inline void wr1(T x){
    print(x);
    putchar(' ');
    return;
}
template<typename T>inline void wr2(T x){
    print(x);
    putchar('\n');
    return;
}
ll n,dp[6100][2],ans;
struct node{
    int x,y;
}a[6100];
inline bool cmp(node A,node B){
    return A.x<B.x;
}
int main(){
    freopen("refract.in","r",stdin);
    freopen("refract.out","w",stdout);
    read(n);
    for(int i=1;i<=n;++i){
        int x,y;
        read(x);
        read(y);
        a[i].x=x;
        a[i].y=y;
    }
    sort(a+1,a+n+1,cmp);
    for(int i=1;i<=n;++i){
        dp[i][0]=dp[i][1]=1;
        for(int j=i-1;j;--j){
            if(a[i].y<a[j].y){
                dp[j][1]=(dp[j][1]+dp[i][0])%mod;
```

```
            }
            else{
                dp[i][0]=(dp[i][0]+dp[j][1])%mod;
            }
        }
    }
    for(int i=1;i<=n;++i){
        ans=(ans+dp[i][0]+dp[i][1]-1)%mod;
    }
    print(ans);
    return 0;
}
```

$T4$

考场：

输出 2!

别问我我为什么，问就是没时间看了。

改题：

$01bfs$。

考虑每个点所在块作为最后下笔的地方，起始代价为 1，对于上下左右可扩展的同色块，扩展代价为 0，异色块扩展代价为 1。

证明：可以先把当前块和异色块先染成一种颜色，然后再染成当前色，显然只用落两笔，没有更优的下笔方法。

找到代价最高的黑色块（一开始全局就是白色块，所以起始笔一定是黑色），记为以当前块为起点的代价，取全局最小值即可。

复杂度 $O(r^2c^2)$。

CODE

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int inf=0x3f3f3f3f;
template<typename T>inline void read(T &x){
    x=0;
    char c=getchar();
    T ret=0;
    while(!isdigit(c))ret|=!(c^'-'),c=getchar();
    while(isdigit(c))x=(x<<3)+(x<<1)+(c^48),c=getchar();
    if(ret)x=(~x)+1;
    return;
}
template<typename T>inline void print(T x){
    if(x<0)putchar('-'),x=(~x)+1;
    if(x>9)print(x/10);
    putchar((x-x/10*10)^48);
    return;
}
```

```cpp
template<typename T>inline void wr1(T x){
    print(x);
    putchar(' ');
    return;
}
template<typename T>inline void wr2(T x){
    print(x);
    putchar('\n');
    return;
}
int ans=inf,r,c,c1[]={0,1,0,-1},c2[]={1,0,-1,0},dis[52][52];
char mp[60][60];
bool vis[52][52];
int bfs01(int sx,int sy){
    int ret=0;
    memset(dis,0x3f,sizeof dis);
    memset(vis,0,sizeof vis);
    deque<pair<int,int>>q;
    q.clear();
    dis[sx][sy]=1;
    q.push_front({sx,sy});
    while(!q.empty()){
        int x=q.front().first;
        int y=q.front().second;
        q.pop_front();
        if(vis[x][y])continue;
        vis[x][y]=1;
        for(int i=0;i<=3;++i){
            int xx=x+c1[i];
            int yy=y+c2[i];
            if(xx>0&&xx<=r&&yy>0&&yy<=c){
                if(mp[x][y]==mp[xx][yy]){
                    if(dis[xx][yy]>dis[x][y]){
                        dis[xx][yy]=dis[x][y];
                        q.push_front({xx,yy});
                    }
                }
                else{
                    if(dis[xx][yy]>dis[x][y]+1){
                        dis[xx][yy]=dis[x][y]+1;
                        q.push_back({xx,yy});
                    }
                }
            }
        }
    }
    for(int i=1;i<=r;++i){
        for(int j=1;j<=c;++j){
            if(mp[i][j]=='1'){
                ret=max(ret,dis[i][j]);
            }
        }
    }
    return ret;
}
```

```
int main(){
    freopen("paint.in","r",stdin);
    freopen("paint.out","w",stdout);
    read(r);
    read(c);
    for(int i=1;i<=r;++i){
        scanf("%s",mp[i]+1);
    }
    for(int i=1;i<=r;++i){
        for(int j=1;j<=c;++j){
            ans=min(ans,bfs01(i,j));
        }
    }
    wr1(ans);
    return 0;
}
```

总结

1. 下次别再同一道题上浪费太多时间好吗，你还没到那种随便想正解的水平，老老实实打暴力。
2. 题目刷少了，补起来，别放弃，既然 AFO 的明天就在眼前，就更应该做好背水一战的准备。