

```

1  /*** Lightoj 1424- New Land:
2  You are given an r(row) x c(column) grid consists of only 0 or 1.
3  Now your task is to find the maximum rectangular area that consists only zeroes(0).
4  Input starts with an integer T ( $\leq 4$ ), denoting the number of test cases.
5  Each case starts with a line containing two integers: r and c ( $1 \leq r, c \leq 2000$ ).
6  Each of the next r lines contains c characters (either 0 or 1) denoting the kingdom.
7  For each case, print the maximum rectangular area that consists only zeroes (0).
8  Input:
9  1
10 5 5
11 10010
12 10001
13 00000
14 10000
15 11010
16  Output: 9
17  ***/
18 #include<bits/stdc++.h>
19 using namespace std;
20 #define mx 2005
21 stack<int>st;
22 char ss[mx][mx];
23 int aa[mx][mx], LL[mx][mx], RR[mx][mx];
24 int main(){
25     int tt; scanf("%d",&tt);
26     for(int ks=1; ks<=tt; ks++){
27         int rr,cc; scanf("%d%d",&rr,&cc);
28         for(int i=1; i<=rr; i++)scanf("%s",ss[i]);
29
30         for(int i=1; i<=rr; i++){
31             for(int j=1; j<=cc; j++){
32                 if(ss[i][j-1]=='1')aa[i][j] = 0;
33                 else aa[i][j] = aa[i-1][j]+1;
34             }
35         }
36
37         for(int i=1; i<=rr; i++){
38             for(int j=1; j<=cc; j++){
39                 while(!st.empty() && aa[i][st.top()]>aa[i][j]){
40                     int id = st.top(); st.pop();
41                     RR[i][id] = j-1;
42                 }
43                 st.push(j);
44             }
45             while(!st.empty()){
46                 int id = st.top(); st.pop();
47                 RR[i][id] = cc;
48             }
49
50             for(int j=cc; j>=1; j--){
51                 while(!st.empty() && aa[i][st.top()]>aa[i][j]){
52                     int id = st.top(); st.pop();
53                     LL[i][id] = j+1;
54                 }
55                 st.push(j);
56             }
57             while(!st.empty()){
58                 int id = st.top(); st.pop();
59                 LL[i][id] = 1;
60             }
61         }
62
63         int ans=0;
64         for(int i=1; i<=rr; i++){
65             for(int j=1; j<=cc; j++){
66                 int area = (RR[i][j]-LL[i][j]+1)*aa[i][j];
67                 ans = max(ans,area);
68             }
69         }
70         printf("Case %d: %d\n",ks,ans);
71     }
72 }
```

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define mx 2005
4 char ss[mx][mx];
5 int aa[mx][mx], tree[4*mx+5];
6 void init(int nd,int b,int e,int id){
7     if(b==e){ tree[nd]=b; return; }
8
9     int lf=nd*2, rg=nd*2+1, md=(b+e)/2;
10
11    init(lf,b,md,id);
12    init(rg,md+1,e,id);
13
14    if(aa[id][tree[lf]]<=aa[id][tree[rg]]) tree[nd]=tree[lf];
15    else tree[nd]=tree[rg];
16}
17 int query(int nd,int b,int e,int x,int y,int id){
18    if(x>e||y<b) return 0;
19    if(b>=x&&e<=y) return tree[nd];
20
21    int lf=nd*2, rg=nd*2+1, md=(b+e)/2;
22
23    int m1=query(lf,b,md,x,y,id);
24    int m2=query(rg,md+1,e,x,y,id);
25
26    if(m1==0) return m2;
27    if(m2==0) return m1;
28    if(aa[id][m1]<=aa[id][m2]) return m1;
29    else return m2;
30}
31 int CalculateMaxArea(int b,int e,int n,int id){
32    if(b<1||e<1||e<b) return 0;
33
34    int maxx=0;
35    int ps=query(1,1,n,b,e,id);
36    int area=(e-b+1)*aa[id][ps];
37    int area2=CalculateMaxArea(b,ps-1,n,id);
38    int area3=CalculateMaxArea(ps+1,e,n,id);
39    return maxx=max(maxx,max(area,max(area2,area3)));
40}
41 int main(){
42    int tt; scanf("%d",&tt);
43    for(int ks=1; ks<=tt; ks++){
44        int rr,cc; scanf("%d%d",&rr,&cc);
45        for(int i=1; i<=rr; i++) scanf("%s",ss[i]);
46
47        for(int i=1; i<=rr; i++){
48            for(int j=1; j<=cc; j++){
49                if(ss[i][j-1]=='1') aa[i][j] = 0;
50                else aa[i][j] = aa[i-1][j]+1;
51            }
52        }
53
54        int ans=0;
55        for(int i=1; i<=rr; i++){
56            init(1,1,cc,i);
57            int area = CalculateMaxArea(1,cc,cc,i);
58            ans = max(ans,area);
59        }
60        printf("Case %d: %d\n",ks,ans);
61    }
62
63    return 0;
}

```