【算法分析】

首先发现棋盘矩阵上横纵坐标之和的奇偶性不同的点都是相反的，所以如果把横纵坐标之和为奇（或者是偶）的点取反（位运算^1），这样任务就变成了求一个最大全0或1的子矩阵。

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

for(int j=1;j<=m;j++)

{

cin>>a[i][j];

if((i+j)&1)

a[i][j]^=1;//反转

}

由于不知道最大的正方形和矩形究竟是全0的还是全1的，所以全0的和全1的都要做一遍。

求最大正方形可以用动规。f[i][j]表示以(i,j)为右下角的最大的正方形的边长。f[i][j]=min(f[i-1][j-1],f[i][j-1],f[i-1][j])+1。

最大矩形的计算，可以用单调栈来做，也可以用悬线法来做。悬线法是目前已知最高效的，它要先预处理出每一点向上的最高高度，再处理出向左向右的最大宽度，这样就能算出面积。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | #include <bits/stdc++.h>  using namespace std;  int n,m;  const int maxn=2000;  int map[maxn+10][maxn+10];  int f[maxn+10][maxn+10],l[maxn+10][maxn+10],r[maxn+10][maxn+10];  int dp0(int x)  {  for (int i=1; i<=n; i++)  for (int j=1; j<=m; j++)  if (map[i][j]==x)  f[i][j]=1;  else  f[i][j]=0;  int temp=0;  for (int i=1; i<=n; i++)  for (int j=1; j<=m; j++)  if (map[i][j] != x)  continue;  else  {  f[i][j]=min(f[i-1][j-1],min(f[i][j-1],f[i-1][j]))+1;  temp=max(temp,f[i][j]);  }  return temp\*temp;  }  int dp1(int x)  {  for (int i=1; i<=n; i++)  for (int j=1; j<=m; j++)  f[i][j]=l[i][j]=r[i][j]=0;  for (int i=1; i<=n; i++)  for (int j=1; j<=m; j++)  if (map[i][j] != x)  continue;  else  f[i][j]=f[i-1][j]+1;  for (int i=1; i<=n; i++)  for (int j=1; j<=m; j++)  if (!f[i][j])  continue;  else  {  l[i][j]=1;  int k=j-1;  while (f[i][j]<=f[i][k])  {  l[i][j]+=l[i][k];  k-=l[i][k];  }  }  for (int i=1; i<=n; i++)  for (int j=m; j>0; j--)  if (!f[i][j])  continue;  else  {  r[i][j]=1;  int k=j+1;  while (f[i][j]<=f[i][k])  {  r[i][j]+=r[i][k];  k+=r[i][k];  }  }  int temp=0;  for (int i=1; i<=n; i++)  for (int j=1; j<=m; j++)  {  int now=(l[i][j]+r[i][j]-1)\*f[i][j];  temp=max(temp,now);  }  return temp;  }  int main()  {  cin>>n>>m;  for (int i=1; i<=n; i++)  for (int j=1; j<=m; j++)  {  map[i][j] = (getchar(), getchar())-48;  if (!((i+j) & 1))  map[i][j]=map[i][j]^1;  }  int ans=max(dp0(0),dp0(1));  printf("%d\n",ans);  ans=max(dp1(0),dp1(1));  printf("%d\n",ans);  fclose(stdin);  fclose(stdout);  return 0;  } |

单调栈参考程序：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | #include <bits/stdc++.h>  #define LL long long  using namespace std;  int a[2005][2005];  int s[2005][2005];  stack<int> S;  int main()  {  int n,m;  scanf("%d%d",&n,&m);  for(int i=1; i<=n; i++)  for(int j=1; j<=m; j++)  scanf("%d",&a[i][j]);  for(int i=1; i<=m; i++)  s[1][i]=1;  for(int i=2; i<=n; i++)  for(int j=1; j<=m; j++)  if(a[i][j]!=a[i-1][j])  s[i][j]=s[i-1][j]+1;  else s[i][j]=1;  int ans=0,ans2=1;  for(int i=1; i<=n; i++)  {  ans=max(ans,s[i][1]);  S.push(1);  for(int j=2; j<=m; j++)  {  ans=max(ans,s[i][j]);  if(a[i][j]!=a[i][j-1])  {  int p=j;  while(!S.empty()&&s[i][j]<=s[i][S.top()])  {  int v=S.top();  S.pop();  ans=max(ans,s[i][v]\*(j-v));  int l=min(s[i][v],j-v);  ans2=max(ans2,l\*l);  s[i][v]=s[i][j];  p=v;  }  S.push(p);  }  else  {  while(!S.empty())  {  int v=S.top();  S.pop();  ans=max(ans,s[i][v]\*(j-v));  int l=min(s[i][v],j-v);  ans2=max(ans2,l\*l);  }  S.push(j);  }  }  while(!S.empty())  {  int v=S.top();  S.pop();  ans=max(ans,s[i][v]\*(m+1-v));  int l=min(s[i][v],m+1-v);  ans2=max(ans2,l\*l);  }  }  printf("%d\n%d",ans2,ans);  return 0;  } |