АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ЛЕДЕНИ КЪСОВЕ

Елементарното решение на тази задача, която вече е била давана на състезание у нас, но с ограничение $N \leq 200$ и за един плътен леден къс е да се симулира процеса на топене. На всяка стъпка от симулирането се определят запълнените с лед клетките на таблицата, които имат 2 или повече празни съседни клетки и тези клетки се „освобождават“ от стопилия се лед. Тъй като броят M на клетките, запълнени с лед, в най-лошия случай е $O(N^2)$, а интуитивното ни очакване за броя на стъпките е, че те са $O(N)$ (което се потвърждава от експериментите), то очакваната сложност на този алгоритъм е $O(N^3)$. Това е твърде много за допусканите в условието стойности на N.

В авторското решение е реализиран алгоритъм, който е построен по класическата схема обхождане в ширина, за графова структура, в която за върхове можем да вземе запълнените в началото с лед клетки, а две клетки да считаме свързани с ребро, ако имат обща страна. За начало на обхождането можем да считаме някакъв фиктивен връх, а първото ниво ще бъде образувано от върховете, които ще се стопят след първия час. Класическата схема не може да бъде приложена директно, тъй като обхождането (стопяването) на един връх не винаги води до обхождане (стопяване) на неговите съседи в следващото ниво на обхождането в ширина. Затова, в този вариант на алгоритъма поддържаме за всеки връх v броя на предпазващите го от разтопяване съседи и обхождането на един от съседите просто намалява този брой. В случай, че с разтопяването на съседа, броят на предпазващите го от разтопяване съседи на v стане точно 2 – тогава можем да обявим v за обходен и да го вкараме в опашката на обхождането. Броят на нивата на обхождане, без нулевото съставено само от фиктивния начален връх, е търсеното време за стопяване на целия леден къс.

Модификациите, които трябва да се направят, не увеличават съществено времето за работа на този алгоритъм и сложността му остава пропорционална на броя на ребрата на графовата структура, които в този случай не са повече от $4M$, което, разбира се е $O(N^2)$.

```
#include <stdio.h>
#define MAXN 5001
char map[MAXN][MAXN],nb[MAXN][MAXN];
struct cell {short int x,y,ct;} cellarr[MAXN*MAXN];
int N,arrb,arre,time;
void push(int x,int y)
{  cellarr[arre].x=x;
   cellarr[arre].y=y;
   cellarr[arre].ct=time+1;
   arre++;
}
int main()
{  int i,j,k;
   scanf("%d",&N);
   for(i=0;i<N;i++)
   {  scanf("%s",map[i]);
      for(j=0;j<N;j++) map[i][j]-='0';
   }
   arrb=arre=time=0;
```

```
   for(i=1;i<N-1;i++)
      for(j=1;j<N-1;j++)
      {  if(map[i][j]!=0)
         {  k=map[i-1][j]+map[i][j-1]+map[i][j+1]+map[i+1][j];
            nb[i][j]=k;
            if(k<=2) push (i,j);
         }
      }
   time=1;
   while(arrb<arre) // като при обх. в ширина
   {  if(cellarr[arrb].ct==time) //topi se v tozi takt
      { int x=cellarr[arrb].x;
        int y=cellarr[arrb].y;
        map[x][y]=0;arrb++; //pop
        if(map[x-1][y]&&nb[x-1][y]>2)
        {  nb[x-1][y]--; if(nb[x-1][y]==2) push(x-1,y);}
        if(map[x][y+1]&&nb[x][y+1]>2)
        {  nb[x][y+1]--; if(nb[x][y+1]==2) push(x,y+1);}
        if(map[x+1][y]&&nb[x+1][y]>2)
        {  nb[x+1][y]--; if(nb[x+1][y]==2) push(x+1,y);}
        if(map[x][y-1]&&nb[x][y-1]>2)
        {  nb[x][y-1]--; if(nb[x][y-1]==2) push(x,y-1);}
      }
      else time++;
   }
   printf("%d\n",time);
   return 0;
}
```

Автор: Красимир Манев