STUACM 第四次集训

2019/11/09

- 暴力美学——枚举的艺术

  Flip Game　　Flip Game 位运算 题解

  The Clocks

  The Troublesome Frog

  Painter's Problem

- 前缀和
  - 一维前缀和
    - 例题：子段求和
    - 原理：$\sum_{i=a}^{b} data_i = \sum_{i=1}^{b} data_i - \sum_{i=1}^{a-1} data_i$
    - 图示：

a_1	a_2	a_3	a_4	a_5	a_6		
a_1	a_1+a_2	a_1+a_2+a_3	a_1+……+a_4	a_1+……+a_5	a_1+……a_6		
a_1	a_2	a_3	a_4	……	a_89526	……	a_100000
a_1	a_1+a_2	a_1+a_2+a_3	a_1+……+a_4	……	a_1+……a_89526	……	a_1+……a_1000000

    - 实现：

```
const int mxn = 1000*1000+10;
int data[mxn];      // 假设有n个数据
int sum[mxn];
void init_sum(int n){
    for(int i=1;i<=n;++i){
        sum[i] = sum[i-1] + data[i];
    }
}
int get_sum(int l, int r){    // 返回[l,r]区间和
    return sum[r]-sum[l-1];
}
```

  - 二维前缀和
    - 例题：
    - 原理：
      $\sum_{i=a}^{b}\sum_{j=c}^{d} data_{ij} = \sum_{i=1}^{b}\sum_{j=1}^{d} data_{ij} - \sum_{i=1}^{b}\sum_{j=1}^{c-1} data_{ij} - \sum_{i=1}^{a-1}\sum_{j=1}^{d} data_{ij} + \sum_{i=1}^{a-1}\sum_{j=1}^{c-1} data_{ij}$

- 图示：

a_1,1	a_1,2	a_1,3	a_1,4
a_2,1	a_2,2	a_2,3	a_2,4
a_3,1	a_3,2	a_3,3	a_3,4
a_1,1	a_1,1+a_1,2	a_1,1+a_1,2+a_1,3	a_1,1+……+a_1,4
a_1,1+ a_2,1	a_1,1+a_1,2+ a_2,1+a_2,2	a_1,1+a_1,2+a_1,3+ a_2,1+a_2,2+a_2,3	a_1,1+……+a_1,4+ a_2,1+……+a_2,4
a_1,1+ a_2,1+ a_3,1	a_1,1+a_1,2+ a_2,1+a_2,2+ a_3,1+a_3,2	a_1,1+a_1,2+a_1,3+ a_2,1+a_2,2+a_2,3+ a_3,1+a_3,2+a_3,3	a_1,1+……+a_1,4+ a_2,1+……+a_2,4+ a_3,1+……+a_3,4

- 实现：

```
const int mxn = 1000+10;
const int mxm = 1000+10;
int data[mxn][mxm];       // 假设有n*m个数据
int sum[mxn][mxm];
void init_sum(int n, int m){
    /*
    for(int i=1;i<=n;++i){
        for(int j=1;j<=m;++j){
            sum[i][j] = data[i][j] + sum[i][j-1];
        }
    }
    for(int j=1;j<=m;++j){
        for(int i=1;i<=n;++i){
            sum[i][j] += sum[i-1][j];
        }
    }
    */
    for(int i=1;i<=n;++i){
        for(int j=1;j<=m;++j){
            sum[i][j] = data[i][j] + sum[i-1][j] + sum[i][j-1] - sum[i-1][j-
1];
        }
    }
}
int get_sum(int ldi, int ldj, int rui, int ruj){    // 返回左下角[ldi,ldj]与右
上角[rui,ruj]所组成矩形的区域和
    return sum[ldi][ruj]-sum[rui-1][ruj]-sum[ldi][ldj-1]+sum[rui-1][ldj-1];
}
```

- 差分数组
  - 一维差分
    - 例题：Color the ball
    - 原理：$data_b = \sum_{i=1}^{b} \Delta data_{i,i-1}$，$data_0 = 0$，$\Delta data_{i,i-1} = data_i - data_{i-1}$

- 图示：

	a_1	a_2	a_3	a_4	a_5	a_6		
0	a_1-0	a_2-a_1	a_3-a_2	a_4-a_3	a_5-a_4	a_6-a_5		
	a_1	a_2	a_3	a_4	·····	a_89526	·····	a_100000
0	a_1-0	a_2-a_1	a_3-a_2	a_4-a_3	·····	a_89526-a_89525	·····	a_1000000-a_999999

- 应用：多次区间修改（一般为同时增或同时减），少量结果访问。

	a_1	a_2	a_3	·····	a_89525	a_89526	·····	a_100000
0	a_1-0	a_2-a_1	a_3-a_2	·····	a_89525-a_89524	a_89526-a_89525	·····	a_1000000-a_999999
	a_1	a_2	a_3+d	·····+d	a_89525+d	a_89526	·····	a_100000
0	a_1-0	a_2-a_1	a_3-a_2+d	a_4-a_3	·····	a_89526-a_89525-d	·····	a_1000000-a_999999

- 实现：

```
const int mxn = 1000*1000+10;
int data[mxn];       // 假设有n个数据
int delta[mxn];
void init_delta(int n){
  for(int i=1;i<=n;++i){
        delta[i] = data[i] - data[i-1];
  }
}
void update_area(int l, int r, int d){     // 区间[l,r]同时增减
    delta[l] += d;
  delta[r+1] -= d;
}
int get_data(int i){     // 返回多次修改后对应的data_i
    int res = 0;
    for(int j=1;j<=i;++j){
        res += delta[j];
    }
    return res;
}
void get_all_data(int n){     // 直接一次性多次区间修改后，当前对应的值覆盖回原数组
  for(int i=1;i<=n;i++){
        data[i] = data[i-1] + dalta[i];
    }
}
```

- 二维差分
  - 例题：Monitor(难度较大)
  - 原理：每一行都看成一维，做一维差分；对于差分结果，每一列再看成一维，再做一维差分。或，行列的处理顺序调转。

- 图示：

	a_1,1	a_1,2	a_1,3	a_1,4
	a_2,1	a_2,2	a_2,3	a_2,4
	a_3,1	a_3,2	a_3,3	a_3,4
0	a_1,1-0	a_1,2-a_1,1	a_1,3-a_1,2	a_1,4-a_1,3
0	a_2,1-0	a_2,2-a_2,1	a_2,3-a_2,2	a_2,4-a_2,3
0	a_3,1-0	a_3,2-a_3,1	a_3,3-a_3,2	a_3,4-a_3,3
0	0	0	0	0
0	(a_1,1-0)-(0)	(a_1,2-a_1,1)-(0)	(a_1,3-a_1,2)-(0)	(a_1,4-a_1,3)-(0)
0	(a_2,1-0)-(a_1,1-0)	(a_2,2-a_2,1)-(a_1,2-a_1,1)	(a_2,3-a_2,2)-(a_1,3-a_1,2)	(a_2,4-a_2,3)-(a_1,4-a_1,3)
0	(a_3,1-0)-(a_2,1-0)	(a_3,2-a_3,1)-(a_2,2-a_2,1)	(a_3,3-a_3,2)-(a_2,3-a_2,2)	(a_3,4-a_3,3)-(a_2,4-a_2,3)

- 应用：多次矩形内区域修改（一般为同时增或同时减），少量结果访问。

	a_1,1	a_1,2	a_1,3	a_1,4
	a_2,1	a_2,2+d	a_2,3+d	a_2,4
	a_3,1	a_3,2+d	a_3,3+d	a_3,4
0	a_1,1-0	a_1,2-a_1,1	a_1,3-a_1,2	a_1,4-a_1,3
0	a_2,1-0	a_2,2-a_2,1+d	a_2,3-a_2,2	a_2,4-a_2,3-d
0	a_3,1-0	a_3,2-a_3,1+d	a_3,3-a_3,2	a_3,4-a_3,3-d
0	0	0	0	0
0	(a_1,1-0)-(0)	(a_1,2-a_1,1)-(0)	(a_1,3-a_1,2)-(0)	(a_1,4-a_1,3)-(0)
0	(a_2,1-0)-(a_1,1-0)	(a_2,2-a_2,1)-(a_1,2-a_1,1)+d	(a_2,3-a_2,2)-(a_1,3-a_1,2)	(a_2,4-a_2,3)-(a_1,4-a_1,3)-d
0	(a_3,1-0)-(a_2,1-0)	(a_3,2-a_3,1)-(a_2,2-a_2,1)	(a_3,3-a_3,2)-(a_2,3-a_2,2)	(a_3,4-a_3,3)-(a_2,4-a_2,3)
0	0	0-d	0	0+d

- 朴素版实现：

```
const int mxn = 1000+10;
const int mxm = 1000+10;
```

```
int data[mxn][mxm];       // 假设有n*m个数据
int delta[mxn][mxm];
void init_delta(int n, int m){
    for(int i=1;i<=n;++i){
        for(int j=1;j<=m;++j){
            delta[i][j] = data[i][j]-data[i][j-1];
        }
    }
    for(int j=1;j<=m;++j){
        for(int i=n;i>=1;--i){
            delta[i][j] -= delta[i-1][j];
        }
    }
}
void update_area(int ldi, int ldj, int rui, int ruj, int d){    // 左下角
[ldi,ldj]和右上角[rui,ruj]所表示的矩形区域同时增减
    delta[rui][ldj] += d;
    delta[ldi+1][ldj] -= d;
    delta[rui][ruj+1] -= d;
    delta[ldi+1][ruj+1] += d;
}
int get_data(int i, int j){    // 返回多次修改后对应的data_ij
    int res = 0;
    for(int ii=1;ii<=i;++ii){
        for(int jj=1;jj<=j;++jj){
            res += delta[ii][jj];
        }
    }
    return res;
}
void get_all_data(int n, int m){    // 直接一次性多次区间修改后，当前对应的值覆盖回
原数组
    for(int j=1;j<=m;++j){    // 按列用前缀和还原到中间状态
        for(int i=1;i<=n;++i){
            delta[i][j] += delta[i-1][j];
        }
    }
    for(int i=1;i<=n;++i){    // 按行用前缀和还原
        for(int j=1;j<=m;++j){
          data[i][j] = data[i]
              [j-1] + delta[i][j];
        }
    }
}
```

- 优美版实现：

  从图示中最终的二维差分数组可以整理出等式：

  $delta[i][j] = data[i][j] - data[i-1][j] - data[i][j-1] + data[i-1][j-1]$

  移项后得另一个等式：

  $data[i][j] = delta[i][j] + data[i-1][j] + data[i][j-1] - data[i-1][j-1]$

```
const int mxn = 1000+10;
const int mxm = 1000+10;
int data[mxn][mxm];       // 假设有n*m个数据
int delta[mxn][mxm];
void init_delta(int n, int m){
    for(int i=1;i<=n;++i){
        for(int j=1;j<=m;++j){
            delta[i][j]=data[i][j]-data[i-1][j]-data[i][j-1]+data[i-1][j-1];
        }
```

```
    }
}
void update_area(int ldi, int ldj, int rui, int ruj, int d){    // 左下角[ldi,ldj]和右上角[rui,ruj]所表示的矩形区域同时增减
    delta[rui][ldj] += d;
    delta[ldi+1][ldj] -= d;
    delta[rui][ruj+1] -= d;
    delta[ldi+1][ruj+1] += d;
}
void get_all_data(int n, int m){    // 直接一次性多次区间修改后，当前对应的值覆盖回原数组
    for(int i=1;i<=n;++i){
        for(int j=1;j<=m;++j){
            data[i][j]=delta[i][j]+data[i-1][j]+data[i][j-1]-data[i-1][j-1];
        }
    }
}
```