

lesson2

高精度

- 存储方式：个位存到数组第0位，最高位存到最后
- 高精度加法
 - $A_i + B_i + t$ (进位0 or 1)

```
1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  vector<int> add(vector<int> &A, vector<int> &B) {
7      vector<int> C;
8
9      for (int i = 0, t = 0; i < A.size() || i < B.size() || t; i++)
10     {
11         if (i < A.size()) t += A[i];
12         if (i < B.size()) t += B[i];
13         C.push_back(t % 10);
14         t /= 10;
15     }
16     return C;
17 }
18
19 int main(void) {
20     string a, b;
21     vector<int> A, B;
22
23     cin >> a >> b;
24
25     for (int i=a.size() - 1; i>=0; i--) A.push_back(a[i] - '0');
26     for (int i=b.size() - 1; i>=0; i--) B.push_back(b[i] -
27     '0');
28
29     auto C = add(A, B);
30
31     for (int i=C.size() - 1; i>=0; i--) printf("%d", C[i]);
32
33     return 0;
34 }
```

- 高精度减法
 - $A_i - B_i - t$
 - $/ \geq 0$ $A_i - B_i - t$
 - $/ < 0$ $A_i - B_i - t + 10$
 - 保证 $A \geq B$
 - $A \geq B$ $A - B$

- $A < B - (B - A)$

```

1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  bool cmp(vector<int> &A, vector<int> &B) {
7      if (A.size() != B.size()) return A.size() > B.size();
8
9      for (int i = A.size() - 1; i >= 0; i--)
10         if (A[i] != B[i])
11             return A[i] > B[i];
12
13         return true;
14     }
15
16     vector<int> sub(vector<int> &A, vector<int> &B) {
17         vector<int> C;
18
19         for (int i = 0, t = 0; i < A.size(); i++) {
20             t = A[i] - t;
21             if (i < B.size()) t -= B[i];
22             C.push_back((t + 10) % 10);
23             if (t < 0) t = 1;
24             else t = 0;
25         }
26
27         while (C.size() > 1 && C.back() == 0) C.pop_back();
28
29         return C;
30     }
31
32     int main(void) {
33         string a, b;
34         vector<int> A, B, C;
35
36         cin >> a >> b;
37
38         for (int i = a.size() - 1; i >= 0; i--) A.push_back(a[i] - '0');
39         for (int i = b.size() - 1; i >= 0; i--) B.push_back(b[i] - '0');
40
41         if (cmp(A, B)) C = sub(A, B);
42         else C = sub(B, A), cout << "-";
43
44         for (int i = C.size() - 1; i >= 0; i--) printf("%d", C[i]);
45
46         return 0;
47     }

```

- 高精度乘法

```

1  #include <iostream>

```

```

2  #include <vector>
3
4  using namespace std;
5
6  vector<int> mul(vector<int> &A, int b) {
7      vector<int> C;
8
9      for (int i = 0, t = 0; i < A.size() || t; i++) {
10         if (i < A.size()) t += A[i] * b;
11         C.push_back(t % 10);
12         t /= 10;
13     }
14
15     while (C.size() > 1 && C.back() == 0) C.pop_back();
16     return C;
17 }
18
19 int main(void) {
20     string a; int b;
21     vector<int> A, C;
22
23     cin >> a >> b;
24     for (int i = a.size() - 1; i >= 0; i--) A.push_back(a[i] -
25 '0');
26
27     C = mul (A, b);
28
29     for (int i = C.size() - 1; i >= 0; i--) printf("%d", C[i]);
30
31     return 0;
32 }

```

- 高精度除法

- 从最高位开始运算

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  using namespace std;
6
7  vector<int> div(vector<int> &A, int &b, int &r) {
8      vector<int> C;
9
10     for (int i = A.size() - 1; i >= 0; i--) {
11         r = r * 10 + A[i];
12         C.push_back(r / b);
13         r %= b;
14     }
15
16     reverse(C.begin(), C.end());
17     while (C.size() > 1 && C.back() == 0) C.pop_back();
18
19     return C;
20 }

```

```

21
22 int main(void) {
23     string a; int b;
24     vector<int> A, C; int r = 0;
25
26     cin >> a >> b;
27     for (int i = a.size() - 1; i >= 0; i--) A.push_back(a[i] -
    '0');
28
29     C = div(A, b, r);
30
31     for (int i = C.size() - 1; i >= 0; i--) printf("%d", C[i]);
32     cout << endl << r;
33
34     return 0;
35 }

```

前缀和&差分 (互为逆运算, 下标从1开始)

前缀和: (下标需要从1开始, 方便定义 s_0 , 处理边界问题, 统一公式形式)

- 前缀和数组: $S_i = a_1 + a_2 + \dots + a_i$
- 如何求 S_i ? 递推一遍

```

1 s[0] = 0
2 for i = 1; i <= n; i++
3     s[i] = s[i-1] + a[i]

```

- 有什么用?
快速求出 $[l, r]$ 的和 $S_r - S_{l-1}$
- 一维前缀和 (求出某段区间和)

```

1 #include <iostream>
2
3 using namespace std;
4
5 const int N = 1e5+10;
6
7 int n, m;
8 int a[N], s[N];
9
10 int main(void) {
11     scanf("%d%d", &n, &m);
12
13     for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
14
15     for (int i = 1; i <= n; i++) s[i] = s[i - 1] + a[i];
16
17     while (m--) {
18         int l, r;

```

```

19         scanf("%d%d", &l, &r);
20         printf("%d\n", s[r] - s[l - 1]);
21     }
22
23     return 0;
24 }

```

- 二维前缀和
- 快速求出某个子矩阵之和
- 左上角 $(x1, y2)$, 右下角 $(x2, y2)$
- 初始化前缀和

```

1 for (i: 1 - n)
2     for (j: 1-m)
3         s[i][j] = s[i-1][j] + s[i][j-1] - s[i-1][j-1] + a[i][j];

```

- S 子矩阵 = $S[x2,y2] - S[x2, y1-1] - S[x1-1, y2] + S[x1-1, y1-1]$

```

1 #include <iostream>
2
3 using namespace std;
4
5 const int N = 1010;
6
7 int n, m, q;
8 int a[N][N], s[N][N];
9
10 int main(void) {
11     scanf("%d%d%d", &n, &m, &q);
12
13     for (int i = 1; i <= n; i++)
14         for (int j = 1; j <= m; j++)
15             scanf("%d", &a[i][j]);
16
17     for (int i = 1; i <= n; i++)
18         for (int j = 1; j <= m; j++)
19             //求前缀和
20             s[i][j] = s[i - 1][j] + s[i][j - 1] - s[i - 1][j - 1] +
a[i][j];
21
22     while (q--) {
23         int x1, y1, x2, y2;
24         scanf("%d%d%d%d", &x1, &y1, &x2, &y2);
25         //利用前缀和求子矩阵和
26         printf("%d\n", s[x2][y2] - s[x2][y1 - 1] - s[x1 - 1][y2] +
s[x1 - 1][y1 - 1]);
27     }
28
29     return 0;
30 }
31 }

```

差分（前缀和的逆运算）：

- 一维差分
 - 构造原数组
 - $b[n] = a[n] - a[n-1]$
 - 初始假定前缀和数组a所有元素为0，通过n次插入操作进行初始化b数组
 - 作用：
 - $O(n)$ $B \rightarrow A$
 - 快速对原数组给定 $[l, r]$ 区间内全部数进行同一种操作（例如加减运算）
 - $b[l] + c, b[r+1] - c$ ($O(1)$ 复杂度)

```
1  #include <iostream>
2
3  using namespace std;
4
5  const int N = 1e5+10;
6
7  int n, m;
8  int a[N], b[N];
9
10 void insert(int l, int r, int c) {
11     b[l] += c;
12     b[r + 1] -= c;
13 }
14
15 int main(void) {
16     scanf("%d%d", &n, &m);
17
18     for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
19
20     for (int i = 1; i <= n; i++) insert(i, i, a[i]);
21
22     while (m--) {
23         int l, r, c;
24         scanf("%d%d%d", &l, &r, &c);
25         insert(l, r, c);
26     }
27
28     for (int i = 1; i <= n; i++) b[i] += b[i - 1], printf("%d
29 ", b[i]);
30
31     return 0;
32 }
```

- 二维差分
- 通过原矩阵 $a[ij]$ 构造差分矩阵 $b[ij]$ ，使得 $a[ij]$ 是 $b[ij]$ 的前缀和
- $b[x, y] += c, b[x+1, y] -= c, b[x, y+1] -= c, b[x+1, y+1] += c$

```
1  #include <iostream>
2
```

```

3  using namespace std;
4
5  const int N = 1010;
6
7  int n, m, q;
8  int a[N][N], b[N][N];
9
10 void insert(int x1, int y1, int x2, int y2, int c) {
11     b[x1][y1] += c;
12     b[x2 + 1][y1] -= c;
13     b[x1][y2 + 1] -= c;
14     b[x2 + 1][y2 + 1] += c;
15 }
16
17 int main(void) {
18     scanf("%d%d%d", &n, &m, &q);
19
20     for (int i = 1; i <= n; i++)
21         for (int j = 1; j <= m; j++)
22             scanf("%d", &a[i][j]);
23
24     for (int i = 1; i <= n; i++)
25         for (int j = 1; j <= m; j++)
26             insert(i, j, i, j, a[i][j]);
27
28     while (q--) {
29         int x1, y1, x2, y2, c;
30         scanf("%d%d%d%d%d", &x1, &y1, &x2, &y2, &c);
31         insert(x1, y1, x2, y2, c);
32     }
33
34     for (int i = 1; i <= n; i++)
35         for (int j = 1; j <= m; j++)
36             b[i][j] += b[i - 1][j] + b[i][j - 1] - b[i - 1][j - 1];
37
38     for (int i = 1; i <= n; i++) {
39         for (int j = 1; j <= m; j++)
40             printf("%d ", b[i][j]);
41         puts("");
42     }
43
44     return 0;
45 }
46

```

