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
>   - Ai+Bi+t(进位0 or 1)
> -
> ```cpp
> #include <iostream>
> #include <vector>
>
> using namespace std;
>
> vector<int> add(vector<int> &A, vector<int> &B) {
>     vector<int> C;
>
>     for (int i = 0, t = 0; i < A.size() || i < B.size() || t; i++)
> {
>               if (i < A.size()) t += A[i];
>         if (i < B.size()) t += B[i];
>         C.push_back(t % 10);
>         t /= 10;
>     }
>
>     return C;
>       }
>
>       int main(void) {
>     string a, b;
>     vector<int> A, B;
>
>     cin >> a >> b;
>
>     for (int i=a.size() - 1; i>=0; i--) A.push_back(a[i] - '0');
>           for (int i=b.size() - 1; i>=0; i--) B.push_back(b[i] -
> '0');
>
>           auto C = add(A, B);
>
>     for (int i=C.size() - 1; i>=0; i--) printf("%d", C[i]);
>
>     return 0;
>       }
> ```
>
> - 高精度减法
>   - Ai-Bi-t
>   - / >=0 Ai-Bi-t
>   - / <0 Ai-Bi-t+10
>   - 保证A >= B
>   - A>=B A-B

- A<B -(B-A)

```
 #include <iostream>
#include <vector>

using namespace std;

 bool cmp(vector<int> &A, vector<int> &B) {
     if (A.size() != B.size()) return A.size() > B.size();

     for (int i = A.size() - 1; i >= 0; i--)
           if (A[i] != B[i])
             return A[i] > B[i];

           return true;
 }

 vector<int> sub(vector<int> &A, vector<int> &B) {
           vector<int> C;

     for (int i = 0, t = 0; i < A.size(); i++) {
           t = A[i] - t;
         if (i < B.size()) t -= B[i];
         C.push_back((t + 10) % 10);
               if (t < 0) t = 1;
         else t = 0;
     }

     while (C.size() > 1 && C.back() == 0) C.pop_back();

     return C;
 }

 int main(void) {
           string a, b;
     vector<int> A, B, C;

       cin >> a >> b;

     for (int i = a.size() - 1; i >= 0; i--) A.push_back(a[i] - '0');
     for (int i = b.size() - 1; i >= 0; i--) B.push_back(b[i] - '0');

     if (cmp(A, B)) C = sub(A, B);
           else C = sub(B, A), cout << "-";

     for (int i = C.size() - 1; i >= 0; i--) printf("%d", C[i]);

     return 0;
 }
```

- 高精度乘法

-
```
#include <iostream>
```

```cpp
#include <vector>

using namespace std;

vector<int> mul(vector<int> &A, int b) {
    vector<int> C;

    for (int i = 0, t = 0; i < A.size() || t; i++) {
        if (i < A.size()) t += A[i] * b;
        C.push_back(t % 10);
        t /= 10;
    }

    while (C.size() > 1 && C.back() == 0) C.pop_back();
    return C;
}

int main(void) {
    string a; int b;
    vector<int> A, C;

    cin >> a >> b;
    for (int i = a.size() - 1; i >= 0; i--) A.push_back(a[i] -
'0');

    C = mul (A, b);

    for (int i = C.size() - 1; i >= 0; i--) printf("%d", C[i]);

    return 0;
}
```

- 高精度除法
    - 从最高位开始运算

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

vector<int> div(vector<int> &A, int &b, int &r) {
    vector<int> C;

    for (int i = A.size() - 1; i >= 0; i--) {
        r = r * 10 + A[i];
        C.push_back(r / b);
        r %= b;
    }

    reverse(C.begin(), C.end());
    while (C.size() > 1 && C.back() == 0) C.pop_back();

    return C;
}
```

```cpp

int main(void) {
    string a; int b;
    vector<int> A, C; int r = 0;

    cin >> a >> b;
    for (int i = a.size() - 1; i >= 0; i--) A.push_back(a[i] -
'0');

    C = div(A, b, r);

    for (int i = C.size() - 1; i >= 0; i--) printf("%d", C[i]);
    cout << endl << r;

    return 0;
}
```

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s0，处理边界问题，统一公式形式）

- 前缀和数组：Si = a1+a2+...+ai
- 如何求Si？ 递推一遍

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

- 有什么用?

  快速求出[l, r]的和 Sr - S(l-1)

- 一维前缀和（求出某段区间和）

```cpp
#include <iostream>

using namespace std;

const int N = 1e5+10;

int n, m;
int a[N], s[N];

int main(void) {
    scanf("%d%d", &n, &m);

    for (int i = 1; i <= n; i++) scanf("%d", &a[i]);

    for (int i = 1; i <= n; i++) s[i] = s[i - 1] + a[i];

    while (m--) {
        int l, r;
        scanf("%d%d", &l, &r);
        printf("%d\n", s[r] - s[l - 1]);
    }

    return 0;
}
```

- 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

```
for (i: 1 - n)
    for (j: 1-m)
        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]

```
#include <iostream>

using namespace std;

const int N = 1010;

int n, m, q;
int a[N][N], s[N][N];

int main(void) {
    scanf("%d%d%d", &n, &m, &q);

    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            scanf("%d", &a[i][j]);

    for (int i = 1; i <= n; i++)
        for (int j = 1;j <= m; j++)
            //求前缀和
            s[i][j] = s[i - 1][j] + s[i][j - 1] - s[i - 1][j - 1] +
a[i][j];

    while (q--) {
        int x1, y1, x2, y2;
        scanf("%d%d%d%d", &x1, &y1, &x2, &y2);
        //利用前缀和求子矩阵和
        printf("%d\n", s[x2][y2] - s[x2][y1 - 1]  - s[x1 - 1][y2] +
s[x1 - 1][y1 - 1]);
    }


    return 0;
}
```

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->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）
  -
```cpp
#include <iostream>

using namespace std;

const int N = 1e5+10;

int n, m;
int a[N], b[N];

void insert(int l, int r, int c) {
    b[l] += c;
    b[r + 1] -= c;
}

int main(void) {
    scanf("%d%d", &n, &m);

    for (int i = 1; i <= n; i++) scanf("%d", &a[i]);

    for (int i = 1; i <= n; i++) insert(i, i, a[i]);

    while (m--) {
        int l, r, c;
        scanf("%d%d%d", &l, &r, &c);
        insert(l, r, c);
    }

    for (int i = 1; i <= n; i++) b[i] += b[i - 1], printf("%d
", b[i]);

    return 0;
}
```

- 二维差分
- 通过原矩阵a[ij]构造差分矩阵b[ij]，使得aij是bij的前缀和
- b[x, y] += c, b[x2+1, y1] -= c, b[x1, y2+1] -=c, b[x2+1, y2+1] += c

```cpp
#include <iostream>

```

```cpp
using namespace std;

const int N = 1010;

int n, m, q;
int a[N][N], b[N][N];

void insert(int x1, int y1, int x2, int y2, int c) {
    b[x1][y1] += c;
    b[x2 + 1][y1] -= c;
    b[x1][y2 + 1] -= c;
    b[x2 + 1][y2 + 1] += c;
}

int main(void) {
    scanf("%d%d%d", &n, &m, &q);

    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            scanf("%d", &a[i][j]);

    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            insert(i, j, i, j, a[i][j]);

    while (q--) {
        int x1, y1, x2, y2, c;
        scanf("%d%d%d%d%d", &x1, &y1, &x2, &y2, &c);
        insert(x1, y1, x2, y2, c);
    }

    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            b[i][j] += b[i - 1][j] + b[i][j - 1] - b[i - 1][j - 1];

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++)
            printf("%d ", b[i][j]);
        puts("");
    }

    return 0;
}

```