tags: 演算法

程式作業3-報告

演算法設計 - 最大值尋找

- 這題與matrix-chain multiplication相似，可以使用**動態規畫(dynamic programing)** 的方式解題

- 假如今天的式子是2-1*10

	0	1	2
0	2	-MAX	-MAX
1	-MAX	1	-MAX
2	-MAX	-MAX	10

  可以像這樣將數字分別填入對角線的對應位置
  s代表目前的範圍(由小到大)
  i代表範圍的開頭數字引數
  j代表範圍的最後一個數字引數
  k會在i到j之間尋訪
  如此一來就可以用k將s範圍內的i~j分成二塊
    - (i ~ k)
    - (k+1 ~ j)

- 需要特別注意的是，k要當成第k個標點符號的兩邊
  例如(i, j, k)，s = 3
    - 如果是(0, 1, 2) ⇒ (2-1)*(10)
    - 如果是(0, 0, 2) ⇒ (2)-(1*10)

- s會從2往上到最大範圍，如此一來就能用DP的方式往上增加範圍直到找到答案

pseudo code - 最大值尋找

```
op[k] #def k-th operator => a op[k] b
M[a, b] #def find maximum from (a to k)(k+1 to b), a ≤ k < b
m[a, b] #def find minimum from (a to k)(k+1 to b), a ≤ k < b

M[a, b] = max(a≤k<b)max(M[a, k] op[k] M[k + 1, b],
                        M[a, k] op[k] m[k + 1, b],
                        m[a, k] op[k] M[k + 1, b],
                        m[a, k] op[k] m[k + 1, b]))

m[a, b] = min(a≤k<b)(min(M[a, k] op[k] M[k + 1, b],
                         M[a, k] op[k] m[k + 1, b],
                         m[a, k] op[k] M[k + 1, b],
                         m[a, k] op[k] m[k + 1, b]))
```

時間複雜度

- s的範圍(2~n) * i的範圍(0~n-s) * k的範圍(i~j, j=i+s-1) * 4種運算方式
  n是數字數量
  $\sum_{s=2}^{n}$(n-s)(s-1)
  = $\sum_{2}^{n}$(ns-n-s^2+s)
  =(n($\frac{n(n+1)}{2}$-1)) - (n(n-1)) - ($\frac{n(n+1)(2n-1)}{6}$-1) + ($\frac{n(n+1)}{2}$-1)
  = $\frac{1}{6}n^3 - \frac{2}{3}n^2 + \frac{2}{3}n$
  = O(n^3)

演算法設計 - 數列加上括號

- 沿用剛才尋找最大值的過程，為最大、最小值分別建立k表及來源，共四個新二維陣列
- 使用類似binary search的遞迴方式，從k表的右上角開始(最大值)往下分解，(i,k)分解成(i, k)和(k+1, j)，直到i==k

pseudo code - 印出括號

trackMax[][] #def maximum come from
trackMin[][] #def minimum come from
eatWhoMax[][] #def left, right sub program need max or min for max
eatWhoMin[][] #def left, right sub program need max or min for min

```
parentString(vector<string> array,int i,int j,int BigSmall)    //global: trackMax[][],
{
    //確認目前要求最大還是最小
    int k;
    if(BigSmall == 1)
        k = trackMax[i][j];
    else if(BigSmall == 2)
        k = trackMin[i][j];

    if(i==j)
        return;

    //k=i或k+1=j時不需要括號
    if(k!=i)
        array add "(" before i;
        array add ")" after k;
    if(k+1!=j)
        array add "(" before k+1;
        array add ")" after j;

    //根據目前最大還是最小，決定下個遞迴取最大或最小
    int left, right, head;
    if(BigSmall == 1)
        head = eatWhoMax[i][j];
    else if(BigSmall == 2)
        head = eatWhoMin[i][j];

    //分配head到left, right
    switch(head)
        case 1:
            left = 1;
            right = 1;
            break;
        case 2:
            left = 1;
            right = 2;
            break;
        case 3:
            left = 2;
            right = 2;
            break;
        case 4:
            left = 2;
            right = 1;
            break;

    ParentString(array, i, k, left);
    ParentString(array, k+1, j, right);
}
```

時間複雜度

- 假設數字有n個
  - 遞迴會到i=j時結束，因此為n次(所有數字都會跑一遍)
  - 每加一次括號就是跑一次遞迴，而括號的範圍是 2 ~ n，因此是n-1次
- 總時間複雜度 = n + n-1 = 2n-1
  = O(n)