2018校隊補選題解

pA. N元一次方程式 Subtask 1

- 呃… …不是就b/a就好了嗎?
- 你說你WA了?
- 你真的有看清楚題目嗎?
- 「等號左右的」絕對誤差和相對誤差，不是x的!
  - 係數很重要
- 例如說$99999x = 1$，如果輸出$x = 0.0000100$一定是錯的
- 用有效位數輸出比較好
  - printf: %.10e / %.10Le； cout:不要加fixed
- WA 5

pA. N元一次方程式 Subtask 2, 3

- 公式解應該沒什麼問題吧?
- WA 22/44

pA. N元一次方程式 Subtask 4

- 高斯消去法$O(N^3)$!
- WA 87

pA. N元一次方程式 Subtask 5

- 為甚麼高斯消去法WA了?
- 講理論不如親自實驗

1	0	0	0	0	0	1	1
100000	1	0	0	0	0	1	2
0	100000	1	0	0	0	1	3
0	0	100000	1	0	0	1	4
0	0	0	100000	1	0	1	5
0	0	0	0	100000	1	1	6
0	0	0	0	0	100000	1	7

- 用double跑跑看，你的程式有沒有輸出奇怪的解?

pA. N元一次方程式 Subtask 5

1	0	0	0	0	0	1	1
100000	1	0	0	0	0	1	2
0	100000	1	0	0	0	1	3
0	0	100000	1	0	0	1	4
0	0	0	100000	1	0	1	5
0	0	0	0	100000	1	1	6
0	0	0	0	0	100000	1	7

- 正確：

```
0.000009999900
0.000019999800
0.000029999700
0.000039999600
0.000049999500
0.000060000100
0.999990000100
```

錯誤：

```
0.000009999900
0.000019999788
0.000030517578
0.000000000000
0.000000000000
-2147483648.000000000000
0.999990000100
```

pA. N元一次方程式 Subtask 5

- 誤差在消去過程中被不斷放大!
- 原因是消去是拿主對角線的值為基準，所以當主對角線上面有很接近0的值就會發生不好的事情
  - 詳情可見數值穩定性；純高斯消去法的數值穩定性非常的差
- 所以要讓主對角線上的數絕對值大一點比較好
- 既然方程組的順序可以任意改變，那就在消去過程中透過把兩橫列交換把最大的換上來吧!
- 這個技巧稱為partial pivoting
- WA 100

pA. N元一次方程式

- 為甚麼還是沒過?
- 一樣來看實例

$$\left[\begin{array}{cccccccc|c}
60000 & 0 & 0 & 0 & 0 & & 1 & 1 \\
-59000 & 59999 & 0 & 0 & 0 & & 1 & 2 \\
-59000 & -59000 & 59998 & 0 & 0 & \cdots & 1 & 3 \\
-59000 & -59000 & -59000 & 59997 & 0 & & 1 & 4 \\
-59000 & -59000 & -59000 & -59000 & 59996 & & 1 & 5 \\
 & & \vdots & & & & \vdots & \vdots \\
-59000 & -59000 & -59000 & -59000 & -59000 & -59000 & 1 & x
\end{array}\right]$$

- 依照這個規律把矩陣放大到100階左右
- 奇怪的數值又出現了

pA. N元一次方程式

- 又是老問題！不過這回極端的值出現在最右邊那排了
- 其實變數的順序好像換了也沒差，最後記得換回來就好
- 於是變成消去前透過交換行、列的方式把當前右下角還沒消去的矩陣中最大的換到當前要操作的那一格來
- 這個技巧稱為full pivoting
- 經過full pivoting的高斯消去法是有數值穩定性保證的
- AC 150

pA. N元一次方程式

- 不想寫full pivoting怎麼辦?
- 拿到矩陣的時候就先random shuffle所有行和列
  - 唬爛作法，但是這樣就沒有辦法構造特殊測資了，所以是好的
- 一樣AC 150

pB. 闖關遊戲 Subtask 1, 2

- 把dp式列出來
- 發現經過一些有點麻煩的預處理之後可以O(1)查詢區間的期望值
- $O(n^2k)$，TLE 22

pB. 闖關遊戲 Subtask 3, 4

- 把dp式展開
- 斜率跟查詢單調
- 斜率優化或CDQ分治
- $O(nk)$，TLE 110

pB. 闖關遊戲

- 考慮不限段數的形況->分成越多段越好
- 每多分一段減少的總期望值隨著總段數減少
- Aliens優化
- $O(nlgC)$，AC 200

pC. 背包問題 Subtask 2, 3

- DFS爆搜吧(?
- 加上適當的剪枝有可能可以過subtask 1, 4
- TLE 30/50/77

pC. 背包問題 Subtask 1-5 (6)

- 這不是就是個每步權重都相同的最短路徑問題嗎？只是節點由當前的背包狀態表示而已
- BFS!
- 開個set紀錄每個狀態有沒有被走過就可以了
- 這個部分實作細節不少，例如背包的內容是無序的所以判相等的時候要小心
- 如果丟unordered_set還可以自己寫hash，賭他不會碰撞，這樣判相等就不用sort了
- TLE 100/125（視實作的常數大小而定）

pC. 背包問題

- 這題一臉沒有好性質可以用，好像也只能BFS爆搜
- 既然要爆搜就爆搜到底吧!
- 輸入總共也只有大約2.7×10^6種，不如全部塞在code裡面如何?
- 本機爆搜!

pC. 背包問題

- 可是如果一個答案平均就算只要0.01秒，也要算2.7×10^4秒，根本搜不完啊?
- 注意到N, M相同的其實可以一起BFS，大幅減少運算時間!
- 實測大概~40mins可以搜完全部
- 要更快的話還可以同時跑好幾個程式，四核心的電腦要10分鐘跑完不是問題

- 不要浪費時間！跑程式的時候就可以寫其他題目了！
  - 記得爆搜前一定要先傳傳看前面的subtask有沒有過，不然浪費一堆時間得到錯的答案就不好了

pC. 背包問題

- Code長度會不會太長?
- TIOJ的限制是5×10^6 bytes，所以只要一個答案用一個字元表示（2.7×10^6 bytes）就可以了
- 可以用英文字母的字串表示
  - 用差分的方式還可以往下壓一半，不過在這裡完全沒有必要
- AC 200

pD. 跑！ Subtask 1

- 卡judge的題目XD

- 呃複製貼上不好嗎?
  - 可是有三個人沒拿到
- WA 1

pD. 跑！ Subtask 2

- 這題其實就是給定在有向森林中問是不是祖先與後代的關係
- 既然所有詢問都在加邊以後，那就在第一次詢問前DFS一次，把進入離開時間記錄好，就可以應付所有詢問了
- $O(Q)$，WA 29

pD. 跑！ Subtask 3, 6

- 建好圖，每次詢問都從a點DFS
- 或者紀錄每個點的父親，每次詢問都從b點往上走
  - 複雜度一樣，但這個做法比較好
- $O(NQ)$，WA 38/39

pD. 跑！ Subtask 4

- 每條邊的起點不同代表形成的一定是一條鍊
- 用treap即可維護
- 加邊代表把兩個treap併起來，查詢的時候就看有沒有在同一棵treap上以及哪一個比較前面
- $O(N + Q \log N)$，WA 32

- 這些subtask併起來剛好100分
  - 題目沒出好，judge也沒有自動併subtask功能，所以併的時候要花一點時間判斷(?

pD. 跑！ Subtask 7

- 這個題目看起來很欠樹鍊剖分?
- 可是邊是動態加的，這樣不就沒辦法維護輕重鍊了?
- 可以!

pD. 跑！ Subtask 7

- 注意到當把兩棵樹連起來的時候，下面的那棵輕重鍊不會改變，而上面那棵則是接點到根路徑上的輕鍊有可能會變成重鍊
- 這樣的邊只有$O(\log N)$條，所以一個一個檢查就好了！
- 詢問的時候則是從b點往上走看能不能走到a點，所以要能快速查詢一個點有沒有在一條鍊上

pD. 跑！ Subtask 7

- 這些操作treap都可以支援!
  - Treap威能
- 輕鍊變重鍊就是一次split加一次merge
- 詢問有沒有在一條鍊上就在treap上往上走到根就可以了
- 決定輕重鍊就要記錄子樹的size，可以用懶人標區間修改完成
- 總複雜度$O(N + Q\log^2 N)$，TLE 95

pD. 跑!

- 回歸最基本的subtask 2做法：看遍歷中進入和離開時間的前後關係
- 遍歷順序有沒有可能動態維護?
- 非常簡單，我們再一次請出treap
- 考慮遍歷順序的序列，則合併兩棵樹就只不過是把下面那棵樹的遍歷序列插入上面那棵樹遍歷序列的對應位置而已
- 另外記錄指向每個點進入和離開的node的指標就可以順利詢問了
  - 葉子的進入和離開只要一個node就好了
- 總複雜度$O(N + Q \log N)$，AC 250
  - 常數有點大，所以時限開到六秒多，被卡常數記得換亂樹種子試試

pD. 跑！ （另解）

- 有沒有辦法像靜態的時候一樣用倍增法找祖先呢?
- 其實可以，只是要做一些調整
- 記錄每一個點的第2^N輩祖先、根和深度
- 再次使用「懶惰」的精神，只有用到的時候才算!
- 每次詢問先更新兩個點a,b的深度d_a,d_b和根
  - 用類似路徑壓縮的方法
- 問完就知道兩個點在不在同一棵樹
- 接下來用2^N輩祖先的組合查b的d_b-d_a祖先是不是a

pD. 跑！ （另解）

- 注意一旦一個點的某個祖先被確定了，它就不會再改變了
  - 另外記錄每個點目前最高被確定的2^N祖先是誰
- 每次詢問某個點的第2^N祖先是誰，就從已經確定的2^N祖先開始倍增
- 總複雜度$O\big((N+Q)\log N\big)$，AC 250
  - 一樣也是因為每個點的2^N祖先只會算一次
  - 常數比平衡樹小很多

pE. Petya and Array

- ~~有沒有覺得題敘跟Codeforces很像~~
  - ~~尤其是你把Codeforces的題敘丟到google翻譯的時候~~
- 把自己的位置跟想去的位置建邊
- 形成很多環
- 發現(?)一個環需要環的大小-1次操作使環上元素歸位
- 輸出n-環數
- $O(n)$，TLE 92/TLE 97/AC 100
  - 時限很緊，可能要壓點常數，例如輸入優化之類的(?)