HKOI Senior Q3 (Desktop Icons) Editorial

Alex Tung
alex20030190@yahoo.com.hk

27 January 2018

Table of Contents

Task Description

- Given N icons on a $R \times C$ desktop. Each icon is either black (B) or white (W) and occupies one slot.

Task Description

- Given N icons on a $R \times C$ desktop. Each icon is either black (B) or white (W) and occupies one slot.
- Each icon also has an importance value v_i.

Task Description

- Given N icons on a $R \times C$ desktop. Each icon is either black (B) or white (W) and occupies one slot.
- Each icon also has an importance value v_i.
- Given K wallpapers. Each wallpaper is black-and-white, with $R \times C$ tiles.

Task Description

- Given N icons on a $R \times C$ desktop. Each icon is either black (B) or white (W) and occupies one slot.
- Each icon also has an importance value v_i.
- Given K wallpapers. Each wallpaper is black-and-white, with $R \times C$ tiles.
- slot_color $=$ icon_color $\rightarrow$ icon hidden;
- slot_color $\neq$ icon_color $\rightarrow$ icon visible.

Task Description

- Given N icons on a $R \times C$ desktop. Each icon is either black (B) or white (W) and occupies one slot.
- Each icon also has an importance value v_i.
- Given K wallpapers. Each wallpaper is black-and-white, with $R \times C$ tiles.
- slot_color $=$ icon_color $\rightarrow$ icon hidden;
- slot_color $\neq$ icon_color $\rightarrow$ icon visible.
- Your task: to maximize the sum of importance values of visible icons.

Task Description

- Given N icons on a $R \times C$ desktop. Each icon is either black (B) or white (W) and occupies one slot.
- Each icon also has an importance value v_i.
- Given K wallpapers. Each wallpaper is black-and-white, with $R \times C$ tiles.
- slot_color $=$ icon_color $\rightarrow$ icon hidden;
- slot_color $\neq$ icon_color $\rightarrow$ icon visible.
- Your task: to maximize the sum of importance values of visible icons.
- You may choose a wallpaper and move the icons around.

Task Description

- Given N icons on a $R \times C$ desktop. Each icon is either black (B) or white (W) and occupies one slot.
- Each icon also has an importance value v_i.
- Given K wallpapers. Each wallpaper is black-and-white, with $R \times C$ tiles.
- slot_color $=$ icon_color $\rightarrow$ icon hidden;
- slot_color $\neq$ icon_color $\rightarrow$ icon visible.
- Your task: to maximize the sum of importance values of visible icons.
- You may choose a wallpaper and move the icons around.
- Perform the following at most 6400 times: choose an icon and move it to an empty slot.

Sample IO

Sample Input 1

```
1 2 2
1 1 W 10
1
WW
WW
```

Sample Output 1

```
0 1 0
```

Sample Input 2

```
1 2 2
1 1 W 10
2
WW
WW
WW
BW
```

Sample Output 2

```
10 2 1
1 1 2 1
```

Sample IO

Sample Input 3

3 2 2
1 1 B 10
1 2 W 10
2 2 B 10
1
BW
WB

Sample Output 3

30 1 3
1 1 2 1
1 2 1 1
2 2 1 2

Sample Output 3b

30 1 0

Constraints

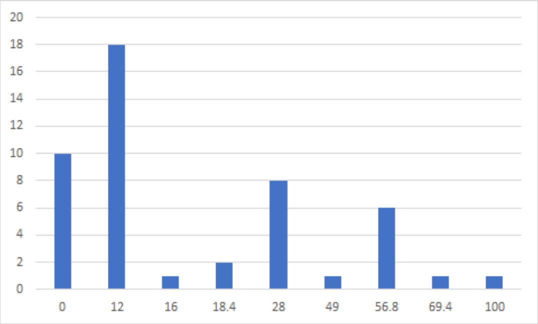
For all cases:
$1 \leq N \leq R \times C$
$1 \leq R, C \leq 80$
$0 \leq v_i \leq 10^5$
$1 \leq K \leq 100$

	Points	Constraints
1	12	$R, C \leq 40$ $K = 1$ All wallpaper tiles are of the same color
2	15	$R, C \leq 40$ $K = 1$
3	16	$R, C \leq 40$ $N = R \times C$
4	21	$R, C \leq 40$ $v_i = 1$
5	12	$R, C \leq 40$
6	24	No additional constraints

- Correct sum and sequence of (at most 6400) moves: 100%
- Correct sum: 40%

Statistics

Attempts: 48
Mean: 21.916
Stddev: 22.272
Top scores: 100 (dbsgame, 1:27), 69.4 (dbscat), 56.8 (6 contestants)
Score distribution:



Comments

- One of the hardest tasks in HKOI Senior

Comments

- One of the hardest tasks in HKOI Senior
- Algorithmically simple, but very hard to code...

Comments

- One of the hardest tasks in HKOI Senior
- Algorithmically simple, but very hard to code...
- Risky to attempt full solution in the beginning

Subtask 1

Subtask 1 (12 points): $R, C \leq 40, K = 1$, tiles of same color

Subtask 1

Subtask 1 (12 points): $R, C \leq 40, K = 1$, tiles of same color

- No need to move icons.

Subtask 1

Subtask 1 (12 points): $R, C \leq 40, K = 1$, tiles of same color

- No need to move icons.
- Just sum the values of all icons, whose color is different from that of the tiles.

Subtask 3

Subtask 3 (16 points): $R, C \leq 40, N = R \times C$

Subtask 3

Subtask 3 (16 points): $R, C \leq 40, N = R \times C$

- Cannot move icons.

Subtask 3

Subtask 3 (16 points): $R, C \leq 40, N = R \times C$

- Cannot move icons.
- For each wallpaper, it is straight-forward to calculate the sum of values of visible icons.

Subtasks 2 and 4

Subtask 2 (15 points): $R, C \leq 40, K = 1$
Subtask 4 (21 points): $R, C \leq 40, v_i = 1$

Subtasks 2 and 4

Subtask 2 (15 points): $R, C \leq 40, K = 1$
Subtask 4 (21 points): $R, C \leq 40, v_i = 1$

- Solving any of these two subtasks is not much easier than solving subtask 5 ($R, C \leq 40$)... It's just easier to code.

Subtasks 5 and 6

Subtask 5 (12 points): $R, C \leq 40$
Subtask 6 (24 points): No additional constraints ($R, C \leq 80$)

Subtasks 5 and 6

Subtask 5 (12 points): $R, C \leq 40$
Subtask 6 (24 points): No additional constraints ($R, C \leq 80$)

- There are two parts to the solution.

Subtasks 5 and 6

Subtask 5 (12 points): $R, C \leq 40$
Subtask 6 (24 points): No additional constraints ($R, C \leq 80$)

- There are two parts to the solution.
- Part 1: to calculate the optimal value and find the wallpaper to be used.

Subtasks 5 and 6

Subtask 5 (12 points): $R, C \leq 40$

Subtask 6 (24 points): No additional constraints ($R, C \leq 80$)

- There are two parts to the solution.
- Part 1: to calculate the optimal value and find the wallpaper to be used.
- Part 2: to find a sequence of moves to achieve the optimal value.

Subtasks 5 and 6

Subtask 5 (12 points): $R, C \leq 40$
Subtask 6 (24 points): No additional constraints ($R, C \leq 80$)

- There are two parts to the solution.
- Part 1: to calculate the optimal value and find the wallpaper to be used.
- Part 2: to find a sequence of moves to achieve the optimal value.
- The intended solutions for subtasks 5 and 6 *differ only in part 2*.

Part 1: finding optimal value and wallpaper

Assumption

Assume that $N < R \times C$.

Part 1: finding optimal value and wallpaper

Assumption

Assume that $N < R \times C$.

- For a wallpaper, we only need to care about the *number* of black (and white) tiles.

Part 1: finding optimal value and wallpaper

Assumption

Assume that $N < R \times C$.

- For a wallpaper, we only need to care about the *number* of black (and white) tiles.
- Suppose that a given wallpaper has B black tiles and W white tiles, and we have b black icons and w white icons.

Part 1: finding optimal value and wallpaper

Assumption

Assume that $N < R \times C$.

- For a wallpaper, we only need to care about the *number* of black (and white) tiles.
- Suppose that a given wallpaper has B black tiles and W white tiles, and we have b black icons and w white icons.
- Clearly we should choose the $\min(b, W)$ black icons and the $\min(w, B)$ white icons of the highest values.

Part 1: finding optimal value and wallpaper

Assumption

Assume that $N < R \times C$.

- For a wallpaper, we only need to care about the *number* of black (and white) tiles.
- Suppose that a given wallpaper has B black tiles and W white tiles, and we have b black icons and w white icons.
- Clearly we should choose the $\min(b, W)$ black icons and the $\min(w, B)$ white icons of the highest values.
- Call these chosen icons *good* and other icons *bad*.

Part 2: finding a sequence of moves

Part 2: finding a sequence of moves

- Now we have chosen a wallpaper and know which icons are good.

Part 2: finding a sequence of moves

- Now we have chosen a wallpaper and know which icons are good.
- We want to make all good icons visible.

Algorithm 1: Icon Swapping

Each time, we "swap" two icons using three moves.

Algorithm 1: Icon Swapping

Each time, we "swap" two icons using three moves.

Pseudo-code

1. while there is a hidden good icon
   Set *GOOD* := the hidden good icon
   Find *EMPTY* := an empty slot
   Find *TARGET* := a tile of opposite color to *GOOD* and not occupied by a **visible** good icon
   Perform *GOOD* $\rightarrow$ *EMPTY*
   Perform *TARGET* $\rightarrow$ *GOOD*
   Perform *EMPTY* $\rightarrow$ *TARGET*
   end

Algorithm 1: Icon Swapping

Each time, we "swap" two icons using three moves.

Pseudo-code

1. while there is a hidden good icon
   Set *GOOD* := the hidden good icon
   Find *EMPTY* := an empty slot
   Find *TARGET* := a tile of opposite color to *GOOD* and not occupied by a **visible** good icon
   Perform *GOOD* $\rightarrow$ *EMPTY*
   Perform *TARGET* $\rightarrow$ *GOOD*
   Perform *EMPTY* $\rightarrow$ *TARGET*
   end

The number of moves is at most $3N$, which solves subtasks 1 - 5.

Algorithm 1: Icon Swapping

Each time, we "swap" two icons using three moves.

Pseudo-code

1. while there is a hidden good icon
   Set *GOOD* := the hidden good icon
   Find *EMPTY* := an empty slot
   Find *TARGET* := a tile of opposite color to *GOOD* and not occupied by a **visible** good icon
   Perform *GOOD* → *EMPTY*
   Perform *TARGET* → *GOOD*
   Perform *EMPTY* → *TARGET*
   end

The number of moves is at most $3N$, which solves subtasks 1 - 5.
Time complexity: $O(NRC)$.

Algorithm 2: Greedily Fix Icons

- For ease of analysis, for each icon (good or bad) we fix a target colour.

Algorithm 2: Greedily Fix Icons

- For ease of analysis, for each icon (good or bad) we fix a target colour.
- At the end, all icons should be placed in a slot having the target colour.

Algorithm 2: Greedily Fix Icons

- For ease of analysis, for each icon (good or bad) we fix a target colour.
- At the end, all icons should be placed in a slot having the target colour.

Pseudo-code

1. while there is a misplaced icon **that can be fixed**
   Let *ICON* := the misplaced icon
   Let *TARGET* := an empty slot having the target colour
   Perform *ICON* → *TARGET*
   end

Each move, we fix exactly one misplaced icon, so the number of moves is at most N. Accepted!

Algorithm 2: Greedily Fix Icons

- For ease of analysis, for each icon (good or bad) we fix a target colour.
- At the end, all icons should be placed in a slot having the target colour.

Pseudo-code

1. while there is a misplaced icon **that can be fixed**
   Let $ICON$:= the misplaced icon
   Let $TARGET$:= an empty slot having the target colour
   Perform $ICON \rightarrow TARGET$
   end

Each move, we fix exactly one misplaced icon, so the number of moves is at most N. Accepted!
Time complexity: $O(NRC)$.

The End

- Questions?