CS2040C Tut 4

Linear Data Structures

Midsem Test Preparation (*priority*)

Stacking Integers

CS2040 AY1718 Special Sem 4

Stacking Integers

You have 3 stacks, with **N** integers in *s1*.

You want to *sort them* and transfer them to *s3*.

(Top: smallest, Bottom: largest)

The integers are a permutation of 1 to **N**.

Stacking Integers

You only can transfer integers from a lower numbered stack to a higher numbered one.

- S1 → S2
- S1 → S3
- S2 → S3

Stacking Integers

Observation

The integers must enter stack s3 in the correct order.

Why?

There is no way to transfer integers out of s3.

Stacking Integers

Approach

At any time, we can compute which is the next integer to be transferred to s3.

If that integer is at the top of s1 or s2, we can transfer it to s3 immediately.

Stacking Integers

Approach

Otherwise, there is only 1 possible move left:

Transfer S1 → S2.

If that is not possible, then print **False**.

Stacking Integers

Time Complexity Analysis

O(N) or $O(N^2)$?

Generals at War

Past Year CS1020E Question

Generals at War

Things to note:

N = 1 000 000

G ≤ **N**

What can be the best algorithm?

Generals at War

Initially, the formation looks like this.

Can we handle **G = 1** casualty report?

Generals at War

Soldiers 5 to 10 are killed.

Generals at War

Who should close the gap?

Generals at War

Who should close the gap?

- First person to the left?
- First person to the right?

Generals at War

Who should close the gap?

- What about **G > 1**?
- Eg: What if they are already dead :O

Generals at War

Who should close the gap?

- First *surviving* person to the left?
- First *surviving* person to the right?

Generals at War

$O(N^2)$ solution (>50%?):

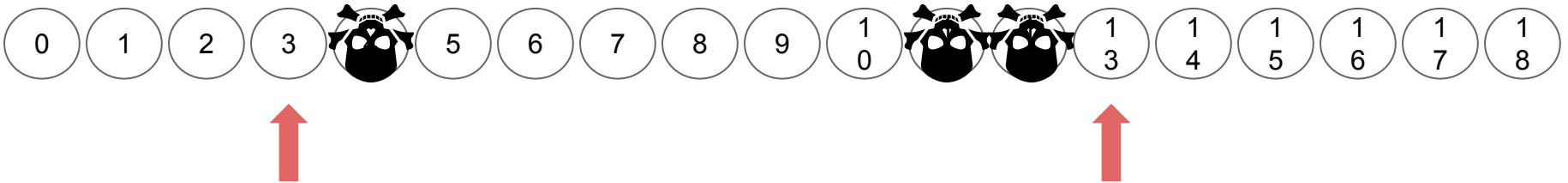
- Loop through to find the first surviving person on both sides

Generals at War

Can we do better?

Once *dead*, always *dead*

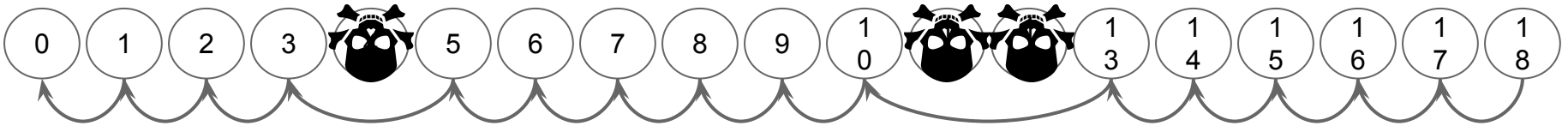
It is quite tiring to keep checking whether a person is dead :O



Generals at War

Lets define 2 things: (for a person *still* alive, **x**)

- alive_left[**x**]: The first person on the left of **x** that is still alive
- alive_right[**x**]: The first person on the right of **x** that is still alive

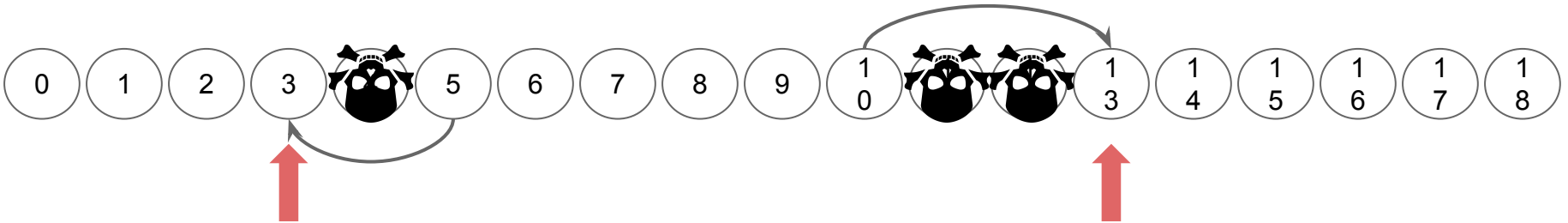


Arrows only show alive_left[x].

Generals at War

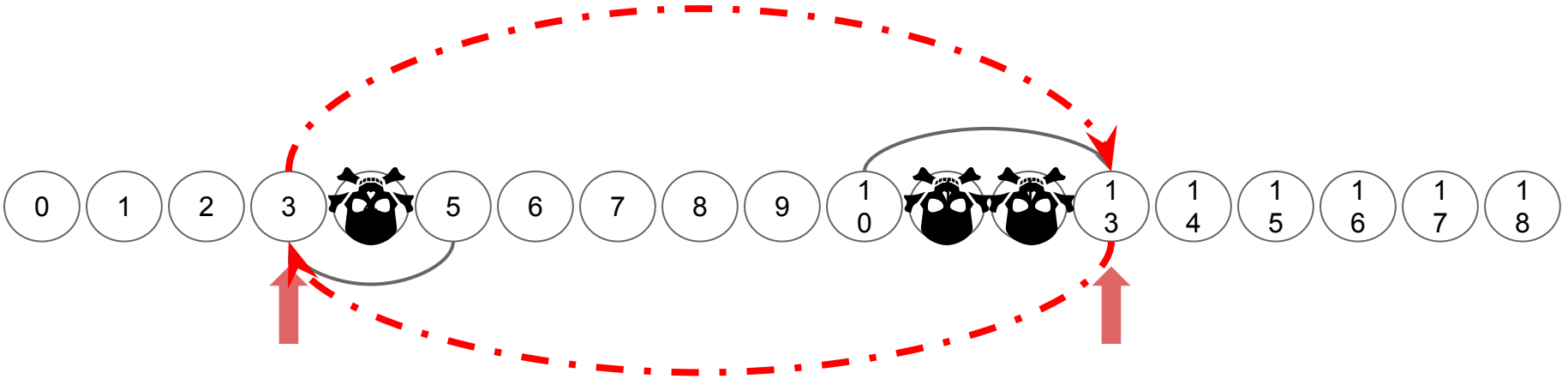
If we have these 2 updated arrays, we can find the 2 surviving soldiers to the left/right quickly.

Time complexity: O(1)



Generals at War

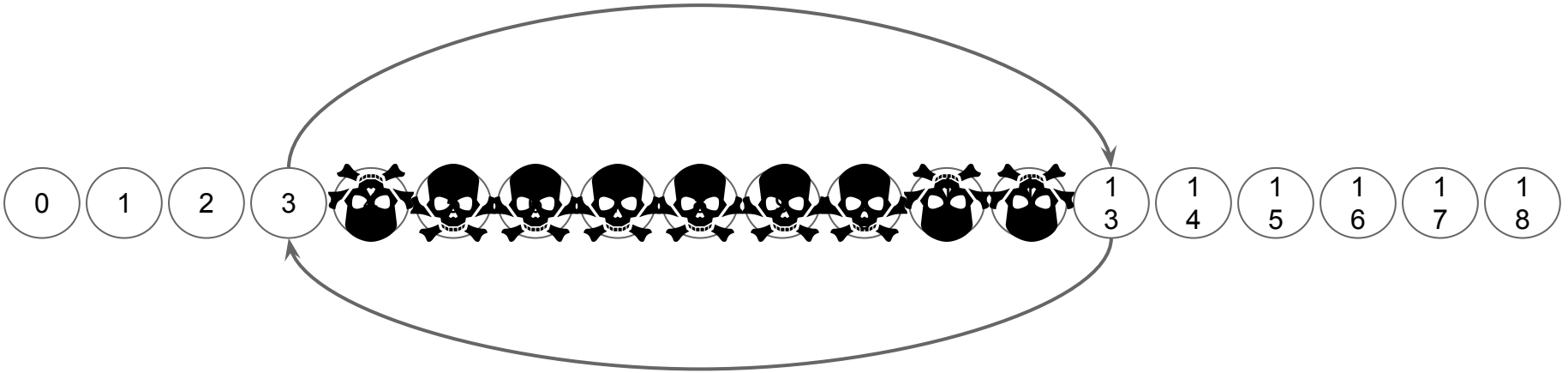
How to keep these 2 arrays updated?



Generals at War

How to keep these 2 arrays updated?

- “Doubly Linked List” style



Generals at War

Implementation Tips/tricks:

- Add dummy soldiers at the two ends
- Always have “somebody alive” on both ends

For practice:

https://uva.onlinejudge.org/external/123/12356.pdf

Generals at War

Past Year Statistics:

70% of 158 students in CS1020E leave the question **blank**.

Advice:

Don’t leave things blank! → Confirm 0 marks

Write *something* → *maybe* some partial marks

Birthday Reminders

Birthday Reminders

We need to sort People in the following way:

- **month** in *ascending* order
- **day** in *ascending* order
- **year** in *descending* order

Birthday Reminders

```
if (a.month < b.month)
    return true;
else if (a.month > b.month)
    return false;
else ...
```

```
if (a.month != b.month)
   return a.month < b.month;
else ...
```

Bracket Matching

Bracket Matching

In lecture we used a stack to keep track of close brackets ‘)’.

Since there’s only (and), what does the stack contain?

Bracket Matching

Observation

The stack only contains the same type of close brackets ‘)’.

We can replace it with an **integer counter** denoting how many of them there are.

Bracket Matching

Approach

counter = 0

If we encounter an open bracket ‘(‘

    counter++

If we encounter a close bracket ‘)’

    counter--

Bracket Matching

Approach

When counter < 0, the brackets do not match.

When counter > 0 at the end, the brackets do not match.

PS2

General Tips

PS2

Tips

- Use List Iterator to keep track of where the cursor is
- List Iterator point to *elements*.
- Cursor points to the area *between characters*.
- Need to decide whether to get the iterator to point to the element *after the cursor* or *before*.

PS2 C/D

- You can add ‘dummy’ characters at the front/back to simplify your code.
- Can add a ‘$’ at the start of your string.
  - If your iterator is to point *before the cursor*.
- Can add a ‘#’ at the end of your string.
  - If your iterator is to point *after the cursor*.

Questions?

All the best for your 15% Midterm Test