# Introduction to Algorithms
# 알고리즘개론
# 2018 Spring Semester

Jinkyu Lee

Dept. of Computer Science and Engineering (and Software),
Sungkyunkwan University (SKKU)

# Rules for all homework

- You should follow instructions.
  - Complier
    - You will get no point if your program cannot be complied with the specified complier
  - Input/output format
    - You will get no point if TA’s automatic evaluation program cannot parse your input or output.
  - Permitted modification scope
    - You will get no point if you modify code outside of the permitted modification scope
  - All other rules
    - You will get severe penalty or no point if you violate the given rules.

# Complier and input/output rules for all homework

- Every implementation homework will be evaluated by TA’s automatic evaluation program with the following complier.
  - Complier: GCC 6.3
  - You will get no point if your program cannot be complied with GCC 6.3.
  - You can use standard library such as *stdlib.h* and *math.h*.

- Input/output format
  - You will get no point if TA’s automatic evaluation program cannot parse your input or output according to the following rules.
  - Use stdin and stdout
  -
    -
  -
    -

- Recommended development environment (Windows)
  - IDE: CodeBlocks (http://www.codeblocks.org/downloads/26)
  - Compiler: MinGW (https://sourceforge.net/projects/mingw)
  - You can use the corresponding compliers for Linux and Mac.

# Homework 3

- 7.5 points (7.5%)
  - 3A: 1.5 points (1.5%)
  - 3B: 2.5 points (2.5%)
  - 3C: 3.5 points (3.5%)
- Due data: 2018/5/21 Monday 23:59
  - Delay penalty: 1% per hour
  - Delay and evaluation will be applied to each file.
  - TA will only evaluate the latest version of your homework with time stamp.
  - Your time management is very important!
- Submission to iCampus
- TA: Jaeheon Kwak
  - 0jaehunny0@gmail.com

# Homework 3

- 3A
  - No file submission
- 3B
  - Code: Yourid_HW3B.c
    - The file type should be c, not cpp.
    - The file should be a single file.
    - Submit to “Homework 3B – Code”
  - Report: Yourid_HW3B.hwp
    - The file type can be hwp, doc(x) or pdf, not others
    - Submit to “Homework 3B – Report”
- 3C
  - Code: Yourid_HW3C.c
    - The file type should be c, not cpp.
    - The file should be a single file.
    - Submit to “Homework 3C – Code”
  - Report: Yourid_HW3C.hwp
    - The file type can be hwp, doc(x) or pdf, not others
    - Submit to “Homework 3C – Report”

# Homework 3A

- 1.5 points (1.5%)

- You will have a in-class quiz in 5/9 (Wed), 5/14 (Mon) or 5/16 (Wed).
    - The coverage is all contents in Lecture Note 10 and 11.
    - If you have any reasonable possibility to be absent in those days, please tell me as soon as possible.
    - You will get no point if you miss the quiz.

# Homework 3B

- Implement a red-black tree as follows:
  - You will be given 3 files
    - "main. c"
    - "redblacktree. h"
    - "redblacktree. c" (Blank file)
  - You should not modify “main.c” and “redblacktree.h”.
  - Implement 6 functions in "redblacktree. c"
    - void rotate_left(rbt_tree * T, rbt_node * x);
    - void rotate_right(rbt_tree * T, rbt_node * x);
    - rbt_tree * rbt_create();
    - rbt_node * insert(rbt_tree * T, int key);
    - void insert_rbt(rbt_tree * T, rbt_node * z);
    - void insert_rbt_fixup(rbt_tree * T, rbt_node * z);

  - TA will evaluate the above 6 functions separately.

# Homework 3B

- How can I compile them? (example: Linux)
  - $ gcc main. c redblacktree. c – o out

- Replace "redblacktree. c" filename with your student ID and submit it.
  - That is, Yourid_HW3B.c
  - Do not submit "main. c" or "redblacktree. h".

- Pseudo code in the lecture note will be helpful, and you are highly recommended to reference it, but…
  - Be careful of NULL.
  - Be careful of a root node.

- Printing format of print_preorder() and print_inorder().
  - " < %d > " for red nodes.
  - " – %d – " for black nodes.

- You can use any standard libraries.

# Homework 3B

- Input
  - The first line contains a single integer $\boldsymbol{N}$.
  - From second line to $\boldsymbol{N+1}$th line contain $\boldsymbol{N}$ integers, representing the red black tree's node values. Those integers will not be duplicate.
  - Example

```
3           // N
1
2
3
```

- Output
  - The output will contain pre-order and in-order tree traverser result.
  - Use print_preorder() and print_inorder() which are implemented in "main.c"
  - Example

```
<2>-1--3-   // print_preorder()
-1-<2>-3-   // print_inorder()
```

- Constraints
  - $\mathbf{0 \leq \boldsymbol{N} \leq 100}$

# Homework 3B

- Sample input for Example 1

  8

  7

  3

  18

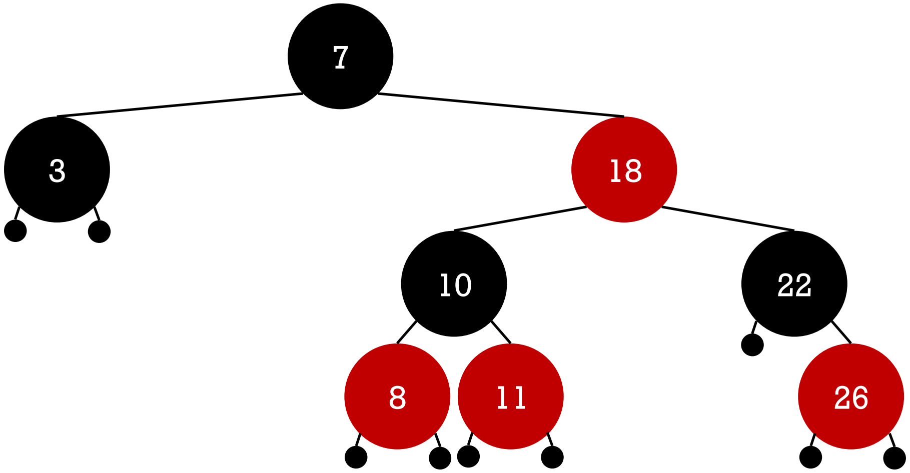
  10

  22

  8

  11

  26



- Sample output for Example 1

  <7><3>-18-<10>-8--11-<22>-26-

  <3><7>-8-<10>-11--18-<22>-26-

# Homework 3B

- Sample input for Example 2

  8

  7

  3

  18

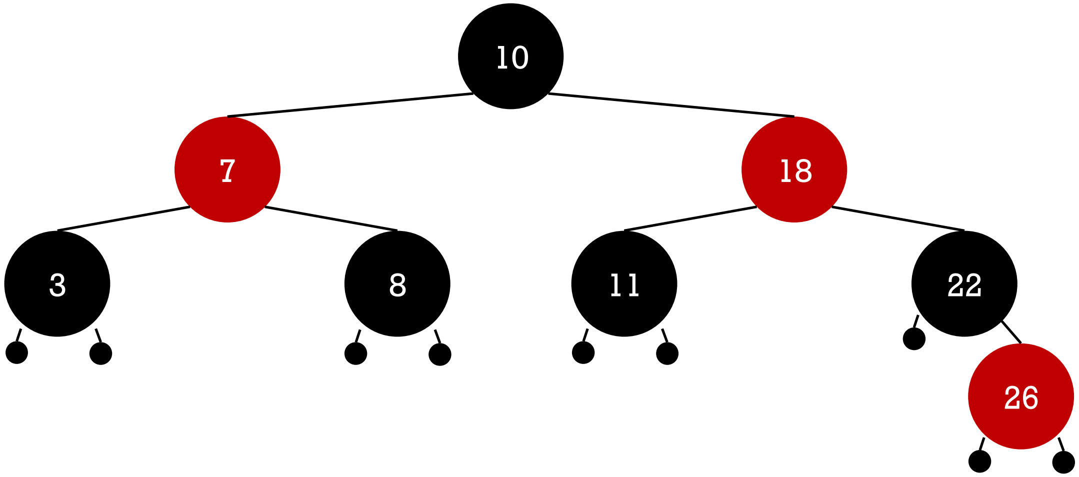
  10

  8

  11

  22

  26



- Sample output for Example 2

  <10>-7-<3><8>-18-<11><22>-26-

  <3>-7-<8><10><11>-18-<22>-26-

# Homework 3B

- Total score: 2.5 points (2.5%)

- Performance evaluation (2.1 points)
  - TA will test several cases.
  - For each case, the result should be printed within 10 seconds.
  - Your C code is tested with the following complier.
    - GCC 6.3
    - You will get zero point if your program cannot be complied with GCC 6.3.
  - You should follow the input and output format.
    - You will get zero point if the TA's automatic evaluation program cannot parse your input or output.

# Homework 3B

- Report evaluation (0.3 points)
  - Explain your code using an example
  - No more than 2 pages
  - In English or Korean

- Code readability (and rules) evaluation (0.1 points)
  - Indent properly
  - Use meaningful names of variables
  - Write sufficient comments in English
    - Do not include any other natural language than English in you code.
  - Use correct file names

# Homework 3C

- Solve a Problem: Sleepy Raccoons
  - You are given a row of $\boldsymbol{N}$ sleepy raccoons, numbered from $\boldsymbol{1}\ \boldsymbol{to}\ \boldsymbol{N}$. You know about each raccoon whether it's sleeping or awake. You can change the state (put to sleep or wake up) of raccoon $\boldsymbol{i}$ if raccoon $\boldsymbol{i+1}$ is awake and raccoons $\boldsymbol{i+2, i+3, \ldots N}$ are sleeping. This rule doesn't apply to raccoon $\boldsymbol{N}$, whose state can be changed at will.
  - Compute the minimum number of changes (changing the state of raccoons), which you need to put all raccoons to sleep.
  - Example

1st

2nd

3rd

4th

- You can change the state of 1st and 4th raccoons.
- You cannot change the state of 2nd raccoon because 3rd raccoon is not awake.
- You cannot change the state of 3rd raccoon because 4th raccoon is not awake.

# Homework 3C

- Input
  - The input contains a single string of values from the set $\{\mathbf{0}, \mathbf{1}\}$. $\boldsymbol{N}$ is equal to the length of the string, and each raccoon is represented by a char. A raccoon that is sleeping is represented by a $\mathbf{0}$, and one that is awake is represented by a $\mathbf{1}$.
  - Example

1

1

0

0

- Output
  - The output should contain a single number representing the minimum number of changes needed.
- Constraints
  - $\mathbf{1} \leq \boldsymbol{N} \leq \mathbf{50}$

# Homework 3C

- Sample input & output 1
  - Input

    00

  - Output

    0

- Sample input & output 2
  - Input

    1100

  - Output

    8

1100 / 0100 / 0101 / 0111 / 0110 / 0010 / 0011 / 0001 / 0000

- Sample input & output 3
  - Input

    1011

  - Output

    13

1011 / 1010 / 1110 / 1111 / 1101 / 1100 / …

# Homework 3C

- Hint 1
  - $\mathbf{1000 \ldots 000}$ of $\boldsymbol{N}$ raccoons need $\mathbf{2}^{\boldsymbol{N}} - \mathbf{1}$ changes to put all them to sleep.
  - You can prove it by induction.
- Hint 2
  - Use dynamic programming with $\boldsymbol{N} * \mathbf{2}$ sized array, $\boldsymbol{dp}[\boldsymbol{N}][\mathbf{2}]$.
  - You can store the minimum number of changes for something.
- Hint 3
  - $\boldsymbol{dp}[\boldsymbol{i}][\mathbf{0}] =$ The minimum number of changes that needed to put to sleep the last $\boldsymbol{N} - \boldsymbol{i} + \mathbf{1}$ raccoons.
  - $\boldsymbol{dp}[\boldsymbol{i}][\mathbf{1}] =$ The minimum number of changes that needed to wake up a $\boldsymbol{i} - \boldsymbol{th}$ raccoon and put to sleep all the following ones.
- Hint 4
  - $\boldsymbol{state}(\boldsymbol{i})$ means the state of a $\boldsymbol{i} - \boldsymbol{th}$ raccoon.
  - **Base case (for the last raccoon)**
  - $\boldsymbol{dp}[\boldsymbol{N}][\boldsymbol{state}(\boldsymbol{N})] = \mathbf{0}$
  - $\boldsymbol{dp}[\boldsymbol{N}][!\,\boldsymbol{state}(\boldsymbol{N})] = \mathbf{1}$

# Homework 3C

- Total score: 3.5 points (3.5%)

- Performance evaluation (3.0 points)
  - TA will test several cases.
  - For each case, the result should be printed within 1 second.
  - Your C code is tested with the following complier.
    - GCC 6.3
    - You will get zero point if your program cannot be complied with GCC 6.3.
  - You should follow the input and output format.
    - You will get zero point if the TA's automatic evaluation program cannot parse your input or output.

# Homework 3C

- Report evaluation (0.4 points)
  - Explain your code using an example
  - No more than 2 pages
  - In English or Korean
- Code readability (and rules) evaluation (0.1 points)
  - Indent properly
  - Use meaningful names of variables
  - Write sufficient comments in English
    - Do not include any other natural language than English in you code.
  - Use correct file names