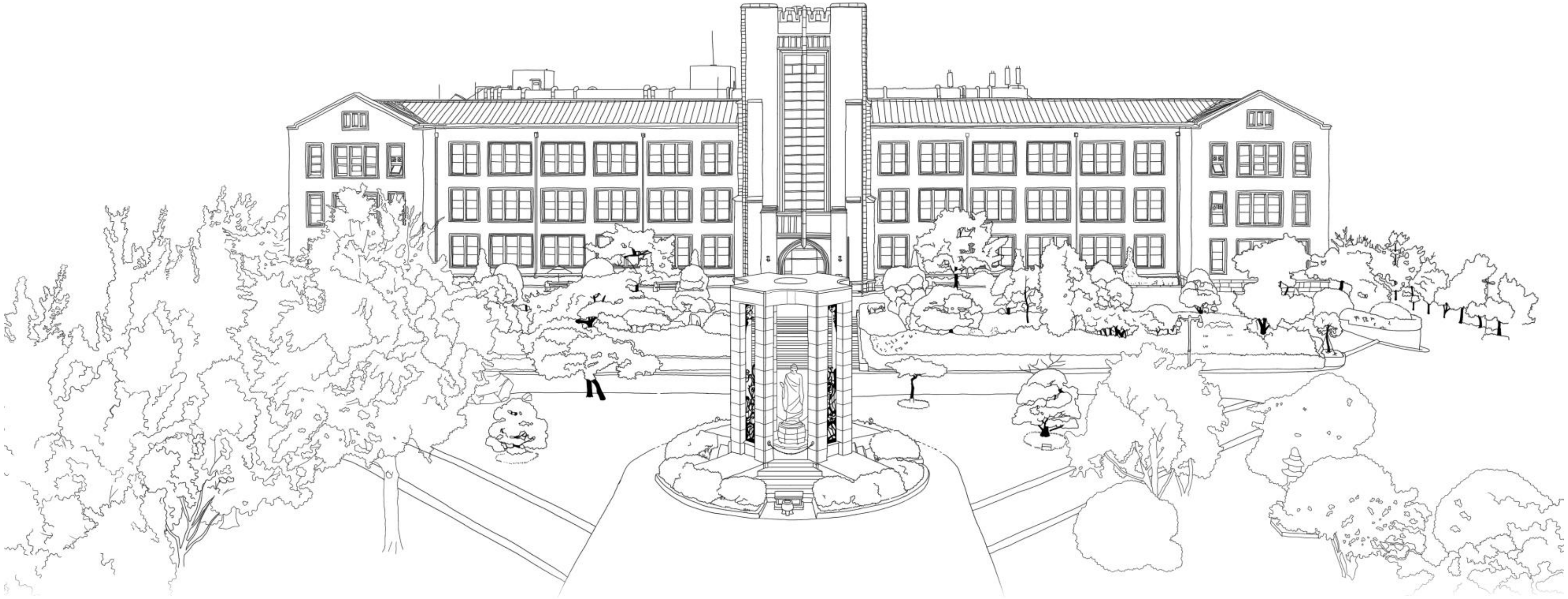


Introduction to Algorithms

DongYeon Kim
Department of Multimedia Engineering
Dongguk University

BigDataLab
Dept. of Multimedia Engineering at Dongguk University

Contents

❖ Chapter.1 Foundations

❖ Proof of the Master theorem

- **Introduction**
  - ➢ Remind the Master theorem
    - ✓ Master method provides a method for solving recurrences of the form $\boldsymbol{T(n) = aT(n/b) + f(n)}$

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \le cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

  - ➢ **To prove the master theorem, we divide in two parts**
    - ✓ First
      i. assume that the $T(n)$ is defined only on exact powers of $b > 1$, that is, $n = 1, b, b^2, \ldots$
      ii. This part gives all the intuition needed to understand why the master theorem is true
    - ✓ Second
      i. show how to extend the analysis to all positive integer n
      ii. Handling floors and ceiling

❖ Proof of the Master theorem

- The proof for exact powers
  - introduction
    - ✓ $\boldsymbol{T(n) = aT(n/b) + f(n)}$
    - ✓ $aT(n/b)$: the cost for conquer step, $f(n)$: the cost for divide & combine step
    - ✓ We will prove under the assumption that n is exact power of $b > 1$
  - We break this analysis into three lemma
    - ✓ First : reduce the problem of solving the master recurrence to the problem of evaluating an expression that contains a summation
    - ✓ Second : determine bounds on this summation
    - ✓ Third : put the first two together to prove a version of the master theorem for the case in which n is an exact poser of b

Divide–and–Conquer

❖Proof of the Master theorem

- The proof for exact powers
  - Step 1 – reduces the problem from master recurrence to the problem contains summation

Lemma 4.2

Let $a \geq 1$ and $b > 1$ be constants, and let $f(n)$ be a nonnegative function defined on exact powers of b. Define $T(n)$ on exact powers of b by the recurrence

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ aT(n/b) + f(n) & \text{if } n = b^i , \end{cases}$$

where i is a positive integer. Then

$$\boxed{T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j) .} \quad (4.21)$$

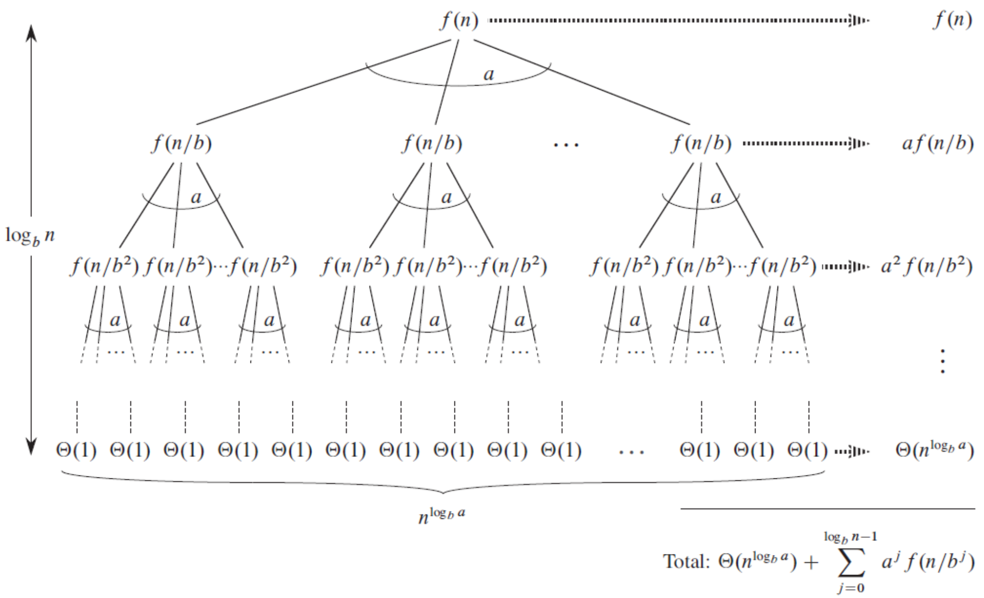
- ✓ $\theta(n^{\log_b a})$: the cost for entire *conquer* step
- ✓ $\sum_{j=0}^{log_b\, n-1} a^j f(n/b^j)$: the cost for entire *divide* & *combine* step

Q : How can we change from $T(n) = aT(n/b) + f(n)$ to $T(n) = \theta(n^{\log_b a}) + \sum_{j=0}^{log_b\, n-1} a^j f(n/b^j)$

A : to prove above change, we use recursion tree

❖ Proof of the Master theorem

- The proof for exact powers
  - ➢ Step 1 – reduces the problem from master recurrence to the problem contains summation



- ✓ Root is $f(n)$ and has a children each with cost $f(n/b)$
- ✓ At dept 2 making a^2 children each with cost $f(n/b^2)$

↓

In general, a^j nodes at depth j, each has cost $f(n/b^j)$

- ✓ The cost of each leaf $T(1) = \theta(1)$
- ✓ Each leaf is at depth $\log_b n$, since $n/b^{\log_b n} = 1$

↓

There are $a^{\log_b n} = n^{\log_b a}$ leaves in the tree

$$\rightarrow T(n) = \underbrace{\theta\left(n^{\log_b a}\right)}_{\text{conquer step cost for } n^{\log_b a} \text{ leaves}} + \underbrace{\sum_{j=0}^{\log_b n-1} a^j f(n/b^j)}_{\text{divide \& combine step cost for every depth except for bottom}}$$

❖ Proof of the Master theorem

- The proof for exact powers
  - ➢ Step 2 – determine bounds on result summation's growth of step 1

Lemma 4.3

Let $a \geq 1$ and $b > 1$ be constants, and let $f(n)$ be a nonnegative function defined on exact powers of b. A function $g(n)$ defined over exact powers of b by

$$g(n) = \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j) \tag{4.22}$$

has the following asymptotic bounds for exact powers of b:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $g(n) = O(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $g(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $a f(n/b) \leq c f(n)$ for some constant $c < 1$ and for all sufficiently large n, then $g(n) = \Theta(f(n))$.

- ✓ Result : $T(n) = \theta\left(n^{\log_b a}\right) + \sum_{j=0}^{log_b\, n-1} a^j f(n/b^j)$
- ✓ $\theta\left(n^{\log_b a}\right)$: asymptotic bounds is not changed
- ✓ $\sum_{j=0}^{log_b\, n-1} a^j f(n/b^j)$: need to determine bounds
- ✓ Define $g(n) = \sum_{j=0}^{log_b\, n-1} a^j f(n/b^j)$

❖ Proof of the Master theorem

- The proof for exact powers
  - ➢ Step 2 – determine bounds on result summation's growth of step 1
    - ✓ For case 1 : If $f(n) = O(n^{\log_b a-\epsilon})$ for some constant $\epsilon > 0$, then $g(n) = O(n^{\log_b a})$

i. $f(n) = O(n^{\log_b a-\epsilon}) \rightarrow f(n/b^j) = O((n/b^j)^{\log_b a-\epsilon})$

ii. $g(n) = \sum_{j=0}^{\log_b n-1} a^j f(n/b^j) \rightarrow g(n) = O(\underline{\sum_{j=0}^{\log_b n-1} a^j (n/b^j)^{\log_b a-\epsilon}})$

iii.
$$\begin{aligned}\sum_{j=0}^{\log_b n-1} a^j \left(\frac{n}{b^j}\right)^{\log_b a-\epsilon} &= n^{\log_b a-\epsilon} \sum_{j=0}^{\log_b n-1} \left(\frac{ab^\epsilon}{b^{\log_b a}}\right)^j \\ &= n^{\log_b a-\epsilon} \sum_{j=0}^{\log_b n-1} (b^\epsilon)^j \\ &= n^{\log_b a-\epsilon} \left(\frac{b^{\epsilon \log_b n}-1}{b^\epsilon-1}\right) = n^{\log_b a-\epsilon} \left(\frac{n^\epsilon-1}{b^\epsilon-1}\right).\end{aligned}$$

iv. Since b and ϵ are constants, we can rewrite the last expression $n^{\log_b a-\epsilon} O(n^\epsilon) = O(n^{\log_b a})$

v. Therefore, we can prove that $g(n) = O(n^{\log_b a})$

❖Proof of the Master theorem

- The proof for exact powers
  - ➢ Step 2 – determine bounds on result summation's growth of step 1
    - ✓ For case 2 : If $f(n) = \theta(n^{\log_b a})$, then $g(n) = \theta(n^{\log_b a} \lg n)$

      i. $f(n) = \theta(n^{\log_b a}) \rightarrow f(n/b^j) = \theta((n/b^j)^{\log_b a})$

      ii. $\mathrm{g}(n) = \sum_{j=0}^{\log_b n-1} a^j f(n/b^j) \rightarrow \mathrm{g}(n) = \theta(\underline{\sum_{j=0}^{\log_b n-1} a^j (n/b^j)^{\log_b a}})$

      iii.
$$\begin{aligned}\sum_{j=0}^{\log_b n-1} a^j \left(\frac{n}{b^j}\right)^{\log_b a} &= n^{\log_b a} \sum_{j=0}^{\log_b n-1} \left(\frac{a}{b^{\log_b a}}\right)^j \\ &= n^{\log_b a} \sum_{j=0}^{\log_b n-1} 1 \\ &= n^{\log_b a} \log_b n\,.\end{aligned}$$

      iv. Therefore, we can prove that $g(n) = \theta(n^{\log_b a} \lg n)$

    - ✓ For case 3 : we can prove same method

❖ Proof of the Master theorem

- The proof for exact powers
  - ➢ Step 3 – combine step 1,2 to prove master theorem

Lemma 4.4

Let $a \ge 1$ and $b > 1$ be constants, and let $f(n)$ be a nonnegative function defined on exact powers of b. Define $T(n)$ on exact powers of b by the recurrence

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ aT(n/b) + f(n) & \text{if } n = b^i , \end{cases}$$

where i is a positive integer. Then $T(n)$ has the following asymptotic bounds for exact powers of b:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \le cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

- ✓ we changed from $T(n) = aT(n/b) + f(n)$ to $T(n) = \theta\left(n^{\log_b a}\right) + \sum_{j=0}^{\log_b n-1} a^j f(n/b^j)$
- ✓ We determined bounds for summation
- ✓ We can now prove the version of master theorem

$$\begin{aligned} T(n) &= \Theta(n^{\log_b a}) + O(n^{\log_b a}) \\ &= \Theta(n^{\log_b a}) , \end{aligned}$$

Case 1

$$\begin{aligned} T(n) &= \Theta(n^{\log_b a}) + \Theta(n^{\log_b a} \lg n) \\ &= \Theta(n^{\log_b a} \lg n) . \end{aligned}$$

Case 2

$$\begin{aligned} T(n) &= \Theta(n^{\log_b a}) + \Theta(f(n)) \\ &= \Theta(f(n)) , \end{aligned}$$

Case 3

❖Proof of the Master theorem

- The proof for all integers
  - ➢ Introduction
    - ✓ To complete the proof of the master theorem, we must now extend analysis to the situation in which floors and ceilings
    - ✓ So that the recurrence is defined for all integers
    - ✓ $T(n) = aT(\lceil n/b \rceil) + f(n)$ or $T(n) = aT(\lfloor n/b \rfloor) + f(n)$
    - ✓ We know $\lceil n/b \rceil \geq n/b$, $\lfloor n/b \rfloor \leq n/b$
  - ➢ Step 1 – reduces the problem
    - ✓ As we go down to recursion tree, we obtain a sequence of recursive invocations on the arguments

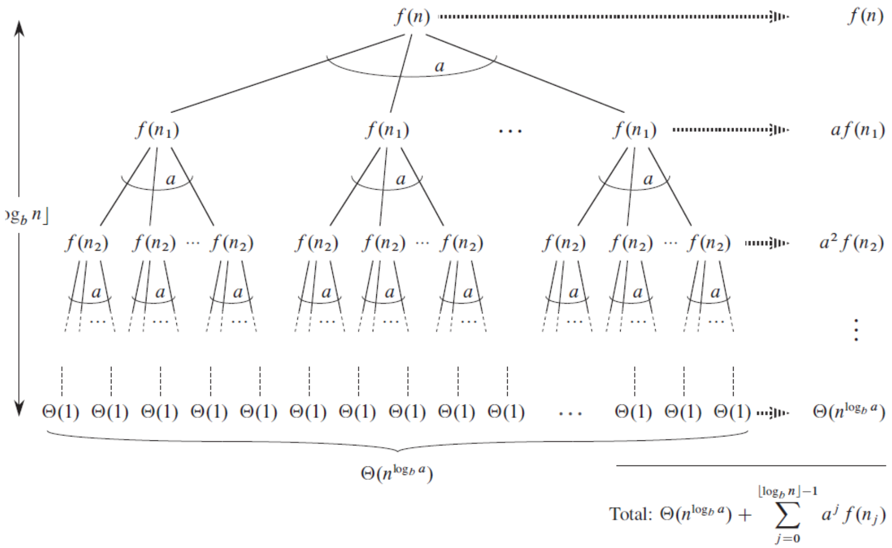
$$
\begin{array}{l}
n\ , \\
\lceil n/b \rceil\ , \\
\lceil \lceil n/b \rceil / b \rceil\ , \\
\lceil \lceil \lceil n/b \rceil / b \rceil / b \rceil\ , \\
\quad\quad \vdots
\end{array}
$$

Let us denote the jth element in the sequence by n_j, where

$$
n_j = \begin{cases} n & \text{if } j = 0\,, \\ \lceil n_{j-1}/b \rceil & \text{if } j > 0\,. \end{cases}
$$

❖Proof of the Master theorem

- The proof for all integers
  - ➢ Step 1 – reduces the problem



✓ At the last depth, j = $\lfloor \log_b n \rfloor$

$$
\begin{aligned}
n_{\lfloor \log_b n \rfloor} &< \frac{n}{b^{\lfloor \log_b n \rfloor}} + \frac{b}{b-1} \\
&< \frac{n}{b^{\log_b n - 1}} + \frac{b}{b-1} \\
&= \frac{n}{n/b} + \frac{b}{b-1} = b + \frac{b}{b-1} = O(1)
\end{aligned}
$$

✓ Our first goal is to determine the k such that n_k is constant

✓ Using $\lceil x \rceil \le x + 1$, we can obtain

$$
\begin{array}{l}
n\ , \\
\lceil n/b \rceil\ , \\
\lceil \lceil n/b \rceil /b \rceil\ , \\
\lceil \lceil \lceil n/b \rceil /b \rceil /b \rceil\ , \\
\vdots
\end{array}
\quad \longrightarrow \quad
\begin{aligned}
n_0 &\le n\ , \\
n_1 &\le \frac{n}{b} + 1\ , \\
n_2 &\le \frac{n}{b^2} + \frac{1}{b} + 1\ , \\
n_3 &\le \frac{n}{b^3} + \frac{1}{b^2} + \frac{1}{b} + 1 \\
&\vdots
\end{aligned}
\quad \longrightarrow \quad
\begin{aligned}
n_j &\le \frac{n}{b^j} + \sum_{i=0}^{j-1} \frac{1}{b^i} \\
&< \frac{n}{b^j} + \sum_{i=0}^{\infty} \frac{1}{b^i} \\
&= \frac{n}{b^j} + \frac{b}{b-1}\ .
\end{aligned}
$$

Node size of each depth — Bounds for each depth — General form

→ we can see that $T(n) = \theta\left(n^{\log_b a}\right) + \sum_{j=0}^{\lfloor \log_b n \rfloor - 1} a^j f(n_j)$

→ step 2, 3 are proved the same way with exact powers

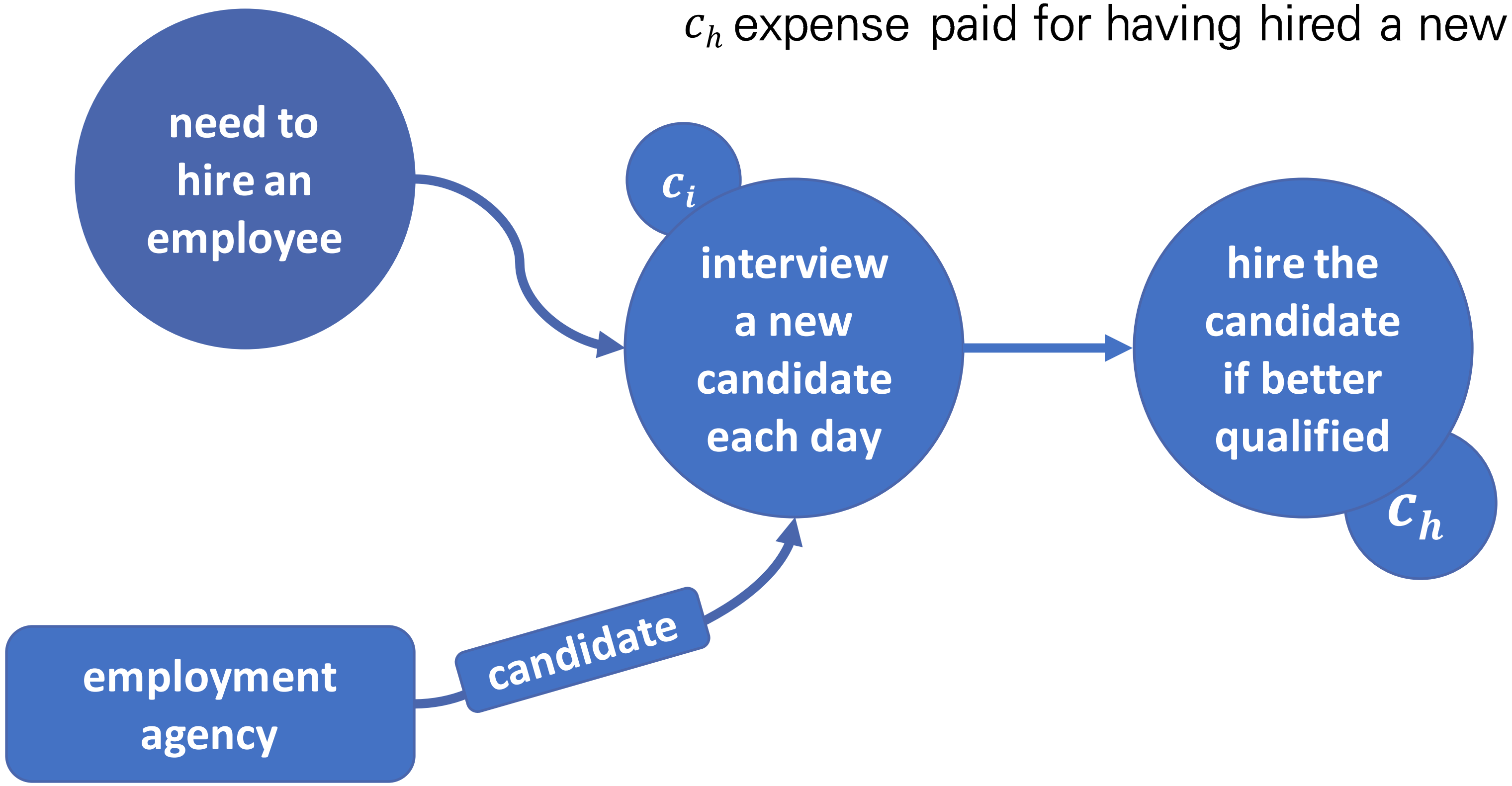
Contents

❖ Chapter.1 Foundations

❖ The hiring problem

- Definition of the hiring problem

c_i expense paid to employment agency for the interviewing candidate
c_h expense paid for having hired a new



❖ The hiring problem

- Pseudocode of the hiring problem
  - Pseudocode

```
HIRE-ASSISTANT(n)
best = 0                // candidate 0 is a least-qualified dummy candidate
for i = 1 to n
    interview candidate i
    if candidate i is better than candidate best
        best = i
        hire candidate i
```

  - We focus on the costs incurred by interviewing and hiring, not the running time
    - ✓ n : the number of candidate, m : the number of hired member
    - ✓ Total cost associated with this algorithm is $O(c_i n + c_h m)$
    - ✓ No matter how many people they hired, always interview n → always incur the cost $c_i n$
  - We have to focus on analyzing $c_h m$
    - ✓ This quantity varies with each run of algorithm

❖ The hiring problem

- **worst case & best case analysis**
  - ➢ Worst case
    - ✓ Hire every candidate that we interview
    - ✓ We hire n times → total hiring cost : $O(c_i n + c_h n)$
  - ➢ Best case
    - ✓ The first candidate has the best quality
    - ✓ we hire 1 times → total hiring cost : $O(c_i n + c_h)$
  - ➢ This is an extreme case that is hard to get up in real.
  - ➢ So, we have to study probability analysis for average case

❖ The hiring problem

- Probabilistic analysis
  - ➢ Probabilistic analysis is the use of probability in the analysis of problem
  - ➢ Commonly it is used to analyze the running time of an algorithm
    - ✓ Sometimes it is used to analyze other quantity (ex. Hiring problem)
  - ➢ We make assumptions about the distribution of inputs before computing average-case running time
    - ✓ We can average the running time over all possible inputs
  - ➢ We must be very careful in deciding on the distribution of inputs
    - ✓ Reasonable assumptions need to be made in order to apply probability analysis

❖ The hiring problem

- Probabilistic analysis for hiring problem
  - ➢ The applicants come in a random order
  - ➢ We can compare any two candidates and decide which is more qualified
    - ✓ They already have a total order on the candidates
    - ✓ We can rank them using $rank(i)$ to denote the rank of applicant i
    - ✓ The order list $< rank(1), rank(2), rank(3), \dots , rank(n) >$
    - ✓ The number of capable order list is $permutation\ of\ (1,2,\dots,n) = n!$ (equally likely)
    - ✓ We have n input samples and the probability is all equal

❖ The hiring problem

- Randomized Algorithms
  - ➢ We know that entire input samples and the probability of each input case occured
  - ➢ but we don't know about input distribution
  - ➢ However we can use probability and randomness as a tool for algorithm design and analysis
    - ✓ Make the behavior of part of the algorithm random.

- Randomized Algorithms in hiring problem
  - ➢ We know the entire case of the order of candidate come & the probability of each case (equally likely)
  - ➢ We can control the algorithm $\boldsymbol{randomized}$ by determining its behavior using $\boldsymbol{random - number\ generator}$ based on probability
  - ➢ $\boldsymbol{random - number\ generator} = \boldsymbol{Random(a, b)}$: return integer between a, b
  - ➢ Now, we can analyze average-case running time using randomized algorithms
  - ➢ When analyzing the running time of a randomized algorithm, we take the expectation of the running time over the distribution of values returned by the random number generator

❖ Indicator random variables

- **Definition of Indicator random variables**
  - ➢ To analyze many algorithms, we use indicator random variables
  - ➢ It provide a convenient method for converting between probabilities and expectation
  - ➢ definition
    - ✓ There are Sample space S, event A
    - ✓ $$\mathrm{I}\{A\} = \begin{cases} 1 & \text{if } A \text{ occurs} , \\ 0 & \text{if } A \text{ does not occur} \end{cases}$$
  - ➢ Example
    - ✓ Filp a fair coin $S = \{H, T\}$, $\Pr\{H\} = \Pr\{T\} = 1/2$
    - ✓ Define an indicator random variable X_H, associated with the coin coming up heads

$$\begin{aligned} X_H &= \mathrm{I}\{H\} \\ &= \begin{cases} 1 & \text{if } H \text{ occurs} , \\ 0 & \text{if } T \text{ occurs} . \end{cases} \end{aligned}$$

$\longrightarrow$ Expected value

$$\begin{aligned} \mathrm{E}[X_H] &= \mathrm{E}[\mathrm{I}\{H\}] \\ &= 1 \cdot \Pr\{H\} + 0 \cdot \Pr\{T\} \\ &= 1 \cdot (1/2) + 0 \cdot (1/2) \\ &= 1/2 . \end{aligned}$$

- ✓ The expected value of an indicator random variable associated with an event A is equal to the probability that A occurs

❖ Indicator random variables

- Definition of Indicator random variables
  - Prove it

$$
\begin{aligned}
\mathrm{E}[X_A] &= \mathrm{E}[\mathrm{I}\{A\}] \\
&= 1 \cdot \Pr\{A\} + 0 \cdot \Pr\{\overline{A}\} \\
&= \Pr\{A\}\ ,
\end{aligned}
$$

where $\overline{A}$ denotes $S - A$, the complement of A.

  - Indicator random variables are useful for analyzing situations in which we perform repeated random trials
  - Example – Calculate the expected value for the number of times the head comes out when n coins are thrown

$$
\begin{aligned}
\mathrm{E}[X] &= \sum_{k=0}^{n} k \cdot \Pr\{X = k\} \\
&= \sum_{k=0}^{n} k \cdot b(k; n, p) \\
&= \sum_{k=1}^{n} k \binom{n}{k} p^k q^{n-k} \\
&= np \sum_{k=1}^{n} \binom{n-1}{k-1} p^{k-1} q^{n-k} \\
&= np \sum_{k=0}^{n-1} \binom{n-1}{k} p^k q^{(n-1)-k} \\
&= np \sum_{k=0}^{n-1} b(k; n-1, p) \\
&= np
\end{aligned}
$$

$\longrightarrow$ Calculate much more easily

$$
\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{n} X_i\right] \\
&= \sum_{i=1}^{n} \mathrm{E}[X_i] \\
&= \sum_{i=1}^{n} p \\
&= np\ .
\end{aligned}
$$

❖ Indicator random variables

- **Indicator random variables for hiring problem**
  - ➢ In hiring problem, we wish to compute the expected number of times that we hire a new assistant
  - ➢ Assume that the candidates arrive in a random order ($randomized\ algorithm$)
  - ➢ X is the random variable, the number of times we hire a new office assistant
  - ➢ The expected value about X
    - ✓ $E[X] = \sum_{x=1}^{n} xPr\{X = x\}$
  - ✓ To calculate expected value more easily, we use indicator random variables

$$X_i = \mathrm{I}\{\text{candidate } i \text{ is hired}\} = \begin{cases} 1 & \text{if candidate } i \text{ is hired ,} \\ 0 & \text{if candidate } i \text{ is not hired ,} \end{cases}$$

and

$$X = X_1 + X_2 + \cdots + X_n .$$

$$\mathrm{E}[X_i] = \Pr\{\text{candidate } i \text{ is hired}\} .$$

$$\longrightarrow$$

$$\mathrm{E}[X_i] = 1/i$$

$$\begin{aligned} \mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{n} X_i\right] && \text{(by equation (5.2))} \\ &= \sum_{i=1}^{n} \mathrm{E}[X_i] && \text{(by linearity of expectation)} \\ &= \sum_{i=1}^{n} 1/i && \text{(by equation (5.3))} \\ &= \ln n + O(1) && \text{(by equation (A.7)) .} \end{aligned}$$

Thank You!