Jacky Ly

①

Algorithms Notes

Feb. 21/21

1.2 Characteristics of Algorithms

1. Input (zero or more)
2. Output (at least one output)
3. Definiteness (every step has a meaning - can be defined)
4. Finiteness (finite set of steps - no infinite loops)
5. Effectiveness (no unnecessary steps)

1.3 Analysis of Algorithms

cost (on slides) { 1. Time - how fast it runs (time function)
2. Space - how much memory it needs (count # of variables and its space) ($1, n, n^2$, etc.)

3. Network consumption
4. Power
5. CPU registers

} not on slides

1 - scalar
n - array
n^2 - matrix

6. Correctness
7. Optimality

} on slides

ex. Fibonacci sequence
1, 1, 2, 3, 5, 8, 13, ...

Correctness ✓
Optimal ?
(From slides)

Cost: $T(n) = \begin{cases} O(1), & n = 1 \text{ or } 2 \\ T(n-1) + T(n-2) + O(1), & \text{otherwise} \end{cases}$

refer to video

```
Fib(n)
  if (n == 1 || n == 2):
      return 1
  otherwise:
      return Fib(n-1) + Fib(n-2)
```

1.4 Frequency Count Method

- regular for loop checks n+1 times
- every line inside loop runs n times
- nested for loop is n(n+1) times
- everything outside loops is 1 (constant)

1.6/1.7 Types of time complexities

1. $O(1)$ - constant
2. $O(\log n)$ - logarithmetic
3. $O(n)$ - linear
4. $O(n^2)$ - quadratic
5. $O(n^3)$ - cubic
6. $O(2^n)$, $O(3^n)$, $O(n^n)$ - exponential

List: $1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \ldots < 2^n < 3^n < \ldots < n^n$

1.8.1/1.8.2 Asymptotic Notations (Big Oh, Omega, Theta)

1. O big-oh — upper bound
2. Ω big-omega — lower bound
3. θ theta — average bound

4. o little-oh
5. ω little-omega

1. Big-oh:

Function $f(n) = O(g(n))$ iff $\exists$ positive constants c and n_0 such that $f(n) \le c \cdot g(n)$ $\forall$ $n \ge n_0$

ex. $f(n) = 2n + 3$ → any coefficient > LHS

$\underbrace{2n+3}_{f(n)} \le \underbrace{10}_{c}\,\underbrace{n}_{g(n)}, \; n \ge 1$ $\quad \therefore f(n) = O(n)$

or

$2n+3 \le 2n + 3n$

$2n+3 \le 5n, \; n \ge 1 \quad \therefore f(n) = O(n)$

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \ldots < 2^n < 3^n < \ldots < n^n$

lower bound — upper bound

average bound

$\therefore f(n) = O(n), O(n^2), O(2^n)$, etc.

all true

closest function (useful)

$f(n) \ne O(\log n), O(\sqrt{n})$, etc.

not true

2. Omega:

Function $f(n) = \Omega(g(n))$ iff $\exists$ positive constants c and n_0 such that $f(n) \ge c \cdot g(n)$ $\forall$ $n \ge n_0$

ex. $f(n) = 2n + 3$

$\underbrace{2n+3}_{f(n)} \ge \underbrace{1}_{c} \cdot \underbrace{n}_{g(n)} \; \forall \; n \ge 1 \quad \therefore f(n) = \Omega(n)$ ↗ useful

$f(n) = \Omega(\log n)$ - lower

but $f(n) \ne \Omega(n^2)$ - upper

Jacky Ly

③

3. Theta

Function $f(n) = \theta(g(n))$ iff $\exists$ positive constants c_1, c_2, and n_0 such that $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

ex. $f(n) = 2n+3$

$$\underbrace{1}_{c_1} \cdot \underbrace{n}_{g(n)} \leq \underbrace{2n+3}_{f(n)} \leq \underbrace{5}_{c_2} \cdot \underbrace{n}_{g(n)}$$

$\therefore f(n) = \theta(n)$ average bound

but $f(n) \neq \theta(n^2)$, $\theta(\log n)$, etc.

Examples:

Ex.1 $f(n) = 2n^2+3n+4$

$2n^2+3n+4 \leq 2n^2+3n^2+4n^2$

$2n^2+3n+4 \leq \underbrace{9}_{c}\,\underbrace{n^2}_{g(n)}$, $n \geq 1$ $\longrightarrow$ $f(n) = O(n^2)$

$2n^2+3n+4 \geq 1 \cdot n^2$ $\longrightarrow$ $f(n) = \Omega(n^2)$

So, $1 \cdot n^2 \leq 2n^2+3n+4 \leq 9n^2$ $\therefore f(n) = \theta(n^2)$

Ex.2 $f(n) = n^2 \log n + n$

$$1 \cdot \underbrace{n^2 \log n}_{g(n)} \leq n^2 \log n + n \leq 10\,\underbrace{n^2 \log n}_{g(n)}$$

$\longrightarrow$ $O(n^2 \log n)$ and $\Omega(n^2 \log n)$ $\therefore \theta(n^2 \log n)$

Ex.3 $f(n) = n! = n \times (n-1) \times (n-2) \times \ldots \times 3 \times 2 \times 1$

$1 \times 1 \times \ldots \times 1 \leq 1 \times 2 \times 3 \times \ldots \times n \leq n \times n \times \ldots \times n$

$1 \leq n! \leq n^n$

$\Omega(1)$ $\quad$ $O(n^n)$ $\quad$ can't find θ (average bound) for $n!$ (f(n))

Ex.4 $f(n) = \log n!$

$\log(1 \times 1 \times \ldots \times 1) \leq \log(1 \times 2 \times 3 \times \ldots \times n) \leq \log(n \times n \times \ldots \times n)$

$1 \leq \log n! \leq \log n^n$

$1 \leq \log n! \leq n \log n$

$\Omega(1)$ $\quad$ $O(n \log n)$

* can't find θ (average bound)

4. Little-oh:

$f(n) \in o(g(n)) \Leftrightarrow \forall c > 0, \exists n_0 > 0$ s.t. $\forall n > n_0$, $f(n) \leq c \cdot g(n)$

g(n) is upper bound for f(n), but grows with different rate

ex. $f(n) \in O(n^3)$, $f(n) \notin \theta(n^3)$

then $f(n) \in o(n^3)$

5. Little-omega:

$f(n) \in \omega(g(n)) \Leftrightarrow \forall c > 0, \exists n_0 > 0$ s.t. $\forall n > n_0$, $f(n) \geq c \cdot g(n)$

g(n) is lower bound for f(n), but grows with different rate

ex. $f(n) \in \Omega(g(n))$, $f(n) \notin \theta(g(n))$

then $f(n) \in \omega(g(n))$

1.9 Properties of Asymptotic Notations

General Properties:

also true for Ω and θ

- if $f(n)$ is $O(g(n))$, then $a \cdot g(n)$ is $O(g(n))$

  ex. $f(n) = 2n^2 + 5$ is $O(n^2)$,

  then $7 \cdot f(n) = 14n^2 + 35$ is also $O(n^2)$

Reflexive:

- if $f(n)$ is given, then $f(n) = O(f(n))$

  ex. $f(n) = n^2$, then it's $O(n^2)$

Transitive:

- if $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$,

  then $f(n) = O(h(n))$

  ex. $f(n) = n$, $g(n) = n^2$, $h(n) = n^3$

  n is $O(n^2)$ and n^2 is $O(n^3)$

  then n is $O(n^3)$

Symmetric:

- if $f(n)$ is $\theta(g(n))$ then $g(n)$ is $\theta(f(n))$

  ex. $f(n) = n^2$, $g(n) = n^2$

  $f(n) = \theta(n^2)$ $g(n) = \theta(n^2)$

Jacky Ly

5

Transpose Symmetric:

- if $f(n) = O(g(n))$ then $g(n)$ is $\Omega(f(n))$

  ex. $f(n) = n$, $g(n) = n^2$

  then n is $O(n^2)$ and

  n^2 is $\Omega(n)$

1. If $f(n) = O(g(n))$ $\quad g(n) \le f(n) \le g(n)$
   and $f(n) = \Omega(g(n))$
   then $f(n) = \Theta(g(n))$

2. If $f(n) = O(g(n))$
   and $d(n) = O(e(n))$
   then $f(n) + d(n) =$
   $O(\max(g(n), e(n)))$
   ↳ max of $g(n)$ and $e(n)$

   Ex. $f(n) = n = O(n)$
   $d(n) = n^2 = O(n^2)$
   $f(n) + d(n) = n + n^2 = O(n^2)$

3. If $f(n) = O(g(n))$
   and $d(n) = O(e(n))$
   then $f(n) \cdot d(n) = O(g(n) \cdot e(n))$
   $n^2 \cdot n \qquad n^2 \cdot n = n^3$

2.1.1 Recurrence Relation $T(n) = T(n-1) + 1$

$$T(n) = \begin{cases} 1, & n = 0 \\ T(n-1)+1, & n > 0 \end{cases}$$

Substitution:

$T(n) = T(n-1) + 1$

$T(n-1) = T(n-2) + 1$

$T(n) = [T(n-2)+1] + 1$

$T(n) = T(n-2) + 2$

$T(n) = [T(n-3)+1] + 2$

$T(n) = T(n-3) + 3$

⋮ continue for k times

$T(n)$ —
```
void T(int n)
{
    if (n>0)
    {
        printf("%d", n);   — 1
        T(n-1)             — T(n-1)
    }
}
```

↓

$T(n) = T(n-1) + 1$

⑥

$T(n) = T(n-k) + k$

Assume $n - k = 0 \quad \therefore n = k$

$T(n) = T(n-n) + n$

$T(n) = T(0) + n \quad \leftarrow T(0) = 1$

$T(n) = 1 + n$

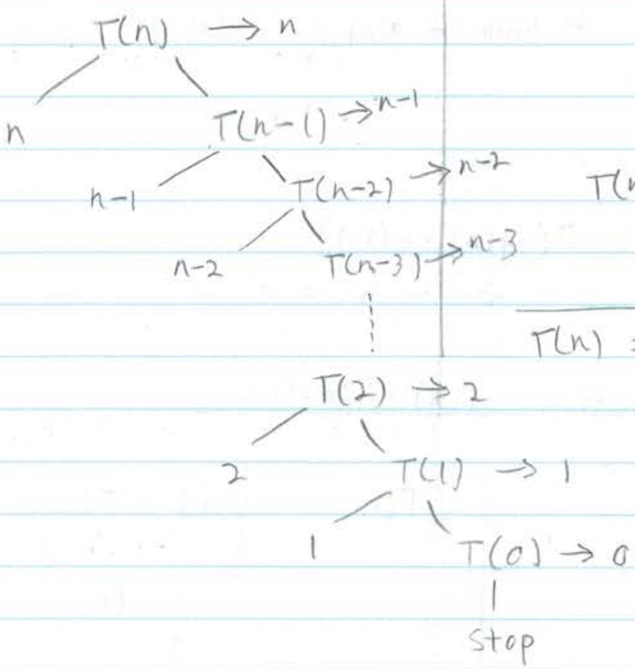
$O(n)$

2.1.2 Recurrence Relation $T(n) = T(n-1) + n$

$T(n) = T(n-1) + n$

$$T(n) = \begin{cases} 1, & n = 0 \\ T(n-1) + n, & n > 0 \end{cases}$$

Recurrence Tree:



$0 + 1 + 2 + \dots + (n-1) + n$

$$T(n) = \frac{n(n+1)}{2} = O(n^2)$$

```
T(n)  — void T(int n)
        {
1     —   if (n>0)
          {
n+1   —     for(i=0; i<n; i++)
            {
n     —        printf("%d", i);
            }
T(n-1) —    T(n-1);
          }
        }
```

$T(n) = T(n-1) + 2n + 2 \leftarrow O(n)$

Jacky Ly

7

$$T(n) = \begin{cases} 1, & n = 0 \\ T(n-1)+n, & n>0 \end{cases}$$

Substitution:

$T(n) = \underline{T(n-1)} + n$ ①

$\because T(n) = T(n-1)+n$

$\therefore \underline{T(n-1)} = T(n-2)+n-1$

$\therefore \underline{T(n-2)} = T(n-3)+n-2$

$T(n) = [T(n-2)+n-1]+n$

$\quad = \underline{T(n-2)} + (n-1)+n$ ②

$T(n) = [T(n-3)+n-2]+(n-1)+n$

$T(n) = T(n-3)+(n-2)+(n-1)+n$ ③

$\vdots$

$T(n) = T(n-k)+(n-(k-1))+(n-(k-2))+\dots+(n-1)+n$ ④

$\downarrow$

Assume $n-k=0 \quad \therefore n=k$

$T(n) = T(n-n)+(\cancel{n}-\cancel{n}+1)+(\cancel{n}-\cancel{n}+2)+\dots+(n-1)+n$

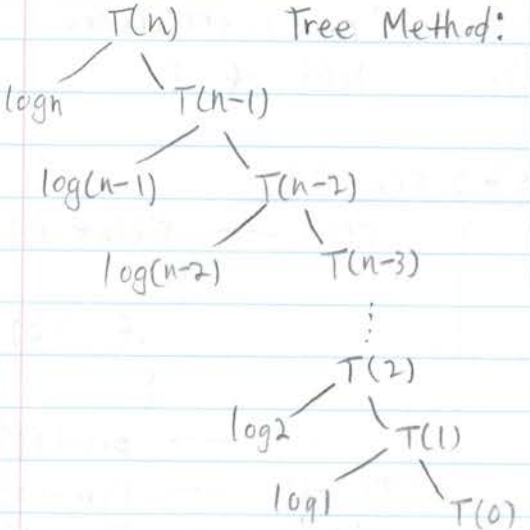
$T(n) = T(0)+\underline{1+2+3+\dots+(n-1)+n}$

$T(n) = 1+\frac{n(n+1)}{2} \searrow O(n^2)$

2.1.3 Recurrence Relation $T(n) = T(n-1) + \log n$

$$T(n) = \begin{cases} 1, & n = 0 \\ T(n-1)+\log n, & n>0 \end{cases}$$

Tree Method:



$T(n)$ —

```
void T(int n)
{
    if(n>0)
    {
        for(i=1, i<n, i=i*2)
        {
            printf("%d", i);      // logn
        }
        T(n-1);                   // T(n-1)
    }
}
```

$T(n) = T(n-1) + \log n$

$\log n + \log(n-1) + \log(n-2) + \dots + \log 2 + \log 1$

$\log[n \cdot (n-1) \cdot \dots \cdot 2 \cdot 1]$

$\log n!$

$O(n \log n)$

Substitution:

$$T(n) = \begin{cases} 1 & , n = 0 \\ T(n-1) + \log n, & n > 0 \end{cases}$$

$T(n) = T(n-1) + \log n$ ①

$T(n) = T(n-2) + \log(n-1) + \log n$ ②

$T(n) = T(n-3) + \log(n-2) + \log(n-1) + \log n$ ③

$\vdots$

$T(n) = T(n-k) + \log 1 + \log 2 + \dots + \log(n-1) + \log n$

Assume $n-k=0$, $\therefore n=k$

$T(n) = T(n-n) + \log n!$

$T(n) = T(0) + \log n!$

$T(n) = 1 + \log n!$ $O(n \log n)$

Examples,

$T(n) = T(n-1) + 1$ — $O(n)$

$T(n) = T(n-1) + n$ — $O(n^2)$

$T(n) = T(n-1) + \log n$ — $O(n \log n)$

$T(n) = T(n-1) + n^2$ — $O(n^3)$

} multiply local cost by n

$T(n) = T(n-2) + 1$ — $\frac{n}{2}$ $O(n)$

$T(n) = T(n-100) + n$ — $O(n^2)$

*no coefficients left of T

2.1.4 Recurrence Relation $T(n) = 2T(n-1) + 1$

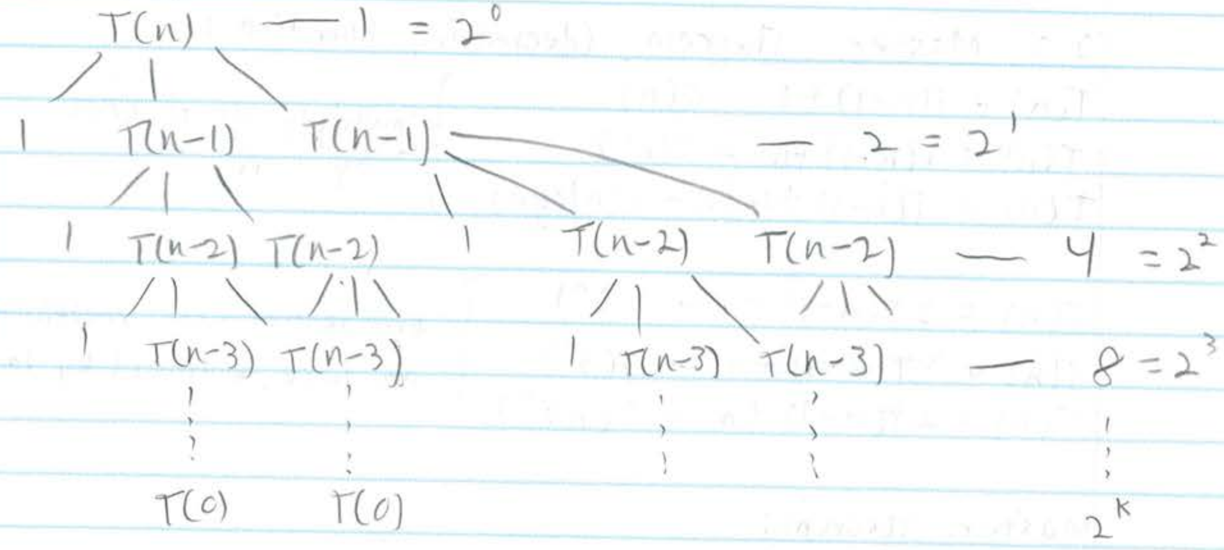
$T(n) = 2T(n-1) + 1$

$$T(n) = \begin{cases} 1 & , n = 0 \\ 2T(n-1) + 1 & , n > 0 \end{cases}$$

```
T(n)   — T(int n)
         {
           if(n>0)
           {
1      —     printf("%d",n);
T(n-1) —     T(n-1);
T(n-1) —     T(n-1);
           }
         }
```

⑨ Jacky Ly

Tree Method:



$1 + 2^1 + 2^2 + 2^3 + \dots + 2^k = 2^{k+1} - 1$

series * $a + ar + ar^2 + ar^3 + \dots + ar^k = \dfrac{a(r^{k+1} - 1)}{r - 1}$

$a = 1 \;;\; r = 2 = \dfrac{1(2^{k+1} - 1)}{2 - 1} = 2^{k+1} - 1$

Assume $n - k = 0 \therefore n = k \qquad = 2^{n+1} - 1 \rightarrow O(2^n)$

Substitution Method:

$T(n) = 2T(n-1) + 1$ ①

$T(n) = 2[2T(n-2) + 1] + 1$

$= 2^2 T(n-2) + 2 + 1$ ②

$= 2^2 [2T(n-3) + 1] + 2 + 1$

$T(n) = 2^3 T(n-3) + 2^2 + 2 + 1$ ③

$\vdots$

$T(n) = 2^k T(n-k) + 2^{k-1} + 2^{k-2} + \dots + 2^2 + 2 + 1$ ④

Assume $n - k = 0 \therefore n = k$

$= 2^n T(0) + \underline{1 + 2 + 2^2 + \dots + 2^{k-1}}$

$= 2^n \cdot 1 + \underline{2^k - 1}$

$= 2^n + 2^n - 1$

$= 2^{n+1} - 1 \rightarrow O(2^n)$

2.2 Master Theorem (decreasing functions)

$T(n) = T(n-1) + 1$ — $O(n)$
$T(n) = T(n-1) + n$ — $O(n^2)$
$T(n) = T(n-1) + \log n$ — $O(n \log n)$
} multiply local cost by n

$T(n) = 2T(n-1) + 1$ — $O(2^n)$
$T(n) = 3T(n-1) + 1$ — $O(3^n)$
$T(n) = 2T(n-1) + n$ — $O(n2^n)$
} exponential with coefficient as base, multiplied by local cost

Master theorem:

$T(n) = aT(n-b) + f(n)$

$a > 0$, $b > 0$ and $f(n) = O(n^k)$ where $k \geq 0$

1. if $a = 1$, $O(n^{k+1})$, $O(n * f(n))$
2. if $a > 1$, $O(n^k a^{n/b})$, $O(f(n)\, a^{n/b})$
3. if $a < 1$, $O(n^k)$, $O(f(n))$

2.3.1 Recurrence Relation – Dividing Function $T(n) = T(\frac{n}{2}) + 1$

$T(n) = T(\frac{n}{2}) + 1$

$$T(n) = \begin{cases} 1 & , n = 1 \\ T(\frac{n}{2}) + 1 & , n > 1 \end{cases}$$

$T(n) \rightarrow 1$

$1 \quad T(\frac{n}{2}) \rightarrow 1$

$1 \quad T(\frac{n}{2^2}) \rightarrow 1$

$1 \quad T(\frac{n}{2^3}) \rightarrow 1$

⋮

k steps $\quad T(\frac{n}{2^k}) \rightarrow 1$ Assume $\frac{n}{2^k} = 1$

$\because \frac{n}{2^k} = 1 \therefore n = 2^k$ and $k = \log_2 n$ $O(\log n)$

$T(n)$ — T(int n)

```
T(int n)
{
    if (n>1)
    {
        printf("%d", n);   — 1
        T(n/2);            — T(n/2)
    }
}
```

11

Jacky Ly

Substitution:

$$T(n) = \begin{cases} 1 & , n = 1 \\ T(\frac{n}{2}) + 1 & , n > 1 \end{cases}$$

$T(n) = T(\frac{n}{2}) + 1$ ①

$\because T(n) = T(\frac{n}{2}) + 1$

$\therefore T(\frac{n}{2}) = T(\frac{n}{2^2}) + 1$

$T(n) = [T(\frac{n}{2^2}) + 1] + 1$

$T(n) = T(\frac{n}{2^2}) + 2$ ②

$T(n) = T(\frac{n}{2^3}) + 3$ ③

$\vdots$

$T(n) = T(\frac{n}{2^k}) + k$ ④ Assume $\frac{n}{2^k} = 1$ $\therefore n = 2^k$ and $k = \log n$

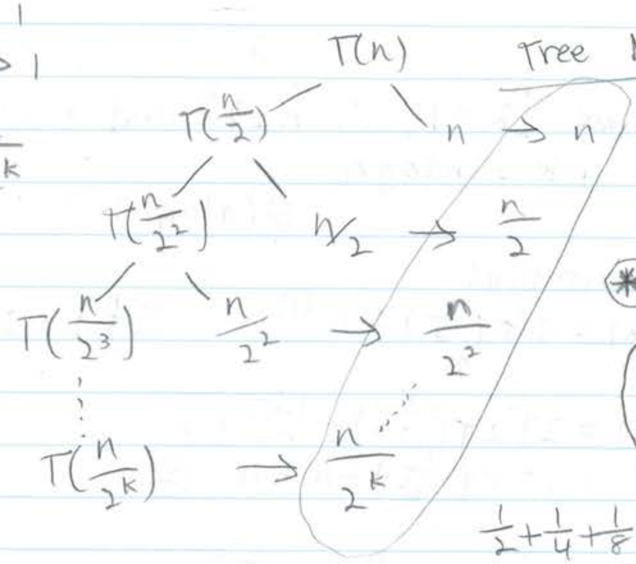
$T(n) = T(1) + \log n$

$T(n) = 1 + \log n \rightarrow O(\log n)$

2.3.2 Recurrence Relation - Dividing Function $T(n) = T(\frac{n}{2}) + n$

$$T(n) = \begin{cases} 1 & , n = 1 \\ T(\frac{n}{2}) + n, & n > 1 \end{cases}$$

Tree Method



$T(n) = n + \frac{n}{2} + \frac{n}{2^2} + \frac{n}{2^3} + \ldots + \frac{n}{2^k}$

$= n\left[1 + \frac{1}{2} + \frac{1}{2^2} + \ldots + \frac{1}{2^k}\right]$

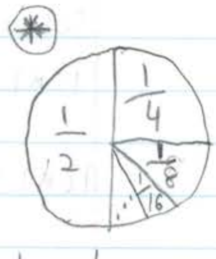
$= n\left(\sum_{i=0}^{k} \frac{1}{2^i}\right) \rightarrow 1$ (*)

$= n \cdot 1$

$T(n) = n$

$O(n)$

(*)



$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \ldots + \infty$

$= 1$ circle

Substitution:

$T(n) = T(\frac{n}{2}) + n$ ①

$T(n) = T(\frac{n}{2^2}) + \frac{n}{2} + n$ ②

$T(n) = T(\frac{n}{2^3}) + \frac{n}{2^2} + \frac{n}{2} + n$

$\vdots$

$T(n) = T(\frac{n}{2^k}) + \frac{n}{2^{k-1}} + \frac{n}{2^{k-2}} + \ldots + \frac{n}{2} + n$

Assume $\frac{n}{2^k} = 1$, $\therefore n = 2^k$ and $k = \log n$

$T(n) = T(1) + n\left[\frac{1}{2^{k-1}} + \frac{1}{2^{k-2}} + \ldots \frac{1}{2} + 1\right]$

$T(n) = 1 + n[1 + 1]$

$T(n) = 1 + 2n$

$O(n)$

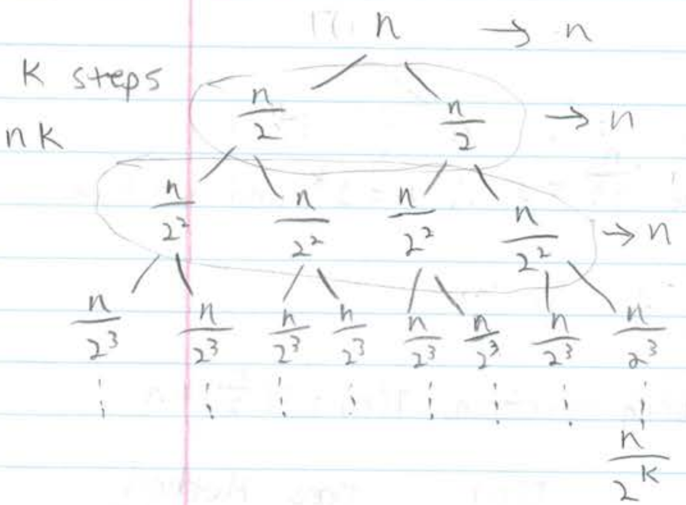
Hilroy

2.3.3 Recurrence Relation - Dividing Function $T(n) = 2T(\frac{n}{2}) + n$

$T(n) = 2T(\frac{n}{2}) + n$

$$T(n) = \begin{cases} 1, & n = 1 \\ 2T(\frac{n}{2}) + n, & n > 1 \end{cases}$$

```
T(n)   —  void T(int n)
          {
              if(n>1)
              {
                for(int i=0; i<n; i++)
                {
n      —        statement;
                }
T(n/2) —        T(n/2);
T(n/2) —        T(n/2);
              }
          }
```



Assume $\frac{n}{2^k} = 1$, $\therefore n = 2^k$ and $k = \log n$

So $nk = n\log n$

$O(n\log n)$

Substitution:

$T(n) = 2T(\frac{n}{2}) + n$ ① $T(\frac{n}{2}) = 2T(\frac{n}{2^2}) + \frac{n}{2}$

$T(n) = 2[2T(\frac{n}{2^2}) + \frac{n}{2}] + n$

$= 2^2 T(\frac{n}{2^2}) + n + n$ ② $T(\frac{n}{2^2}) = 2T(\frac{n}{2^3}) + \frac{n}{2^2}$

$T(n) = 2^2[2T(\frac{n}{2^3}) + \frac{n}{2^2}] + 2n$

$= 2^3 T(\frac{n}{2^3}) + 3n$ ③

$\vdots$

$T(n) = 2^k T(\frac{n}{2^k}) + kn$ Assume $T(\frac{n}{2^k}) = T(1)$

$T(n) = 2^k T(1) + kn$ $\therefore \frac{n}{2^k} = 1$ and $n = 2^k$, $k = \log n$

$= n \cdot 1 + n\log n$

$O(n\log n)$

Jacky Ly

13

2.4.1 Master Theorem (Dividing Functions)

$T(n) = aT(\frac{n}{b}) + f(n)$

$a \geq 1$ $\quad f(n) = \theta(n^k \log^p n)$ $\quad$ ① $\log_b a$

$b > 1$ $\quad$ ② k

case 1: if $\log_b a > k$ then $\theta(n^{\log_b a})$

case 2: if $\log_b a = k$

if $p > -1$ then $\theta(n^k \log^{p+1} n)$

if $p = -1$ then $\theta(n^k \log\log n)$

if $p < -1$ then $\theta(n^k)$

case 3: if $\log_b a < k$

if $p \geq 0$ $\theta(n^k \log^p n)$

if $p < 0$ $\theta(n^k)$

Ex1 $T(n) = 2T(\frac{n}{2}) + 1$

$a = 2$

$b = 2$

$f(n) = \theta(1) = \theta(n^0 \log^0 n)$

$k = 0,\ p = 0$

① $\log_2 2 = 1 >$ ② $k = 0$

satisfies case 1:

$\theta(n^1)$

Ex2 $T(n) = 4T(\frac{n}{2}) + n$

① $\log_2 4 = 2 >$ ② $k = 1,\ p = 0$

case 1: $\theta(n^2)$

Ex3 $T(n) = 8T(\frac{n}{2}) + n^2$

$\log_2 8 = 3 > k = 2$ $\quad$ case 1: $\theta(n^3)$

Ex4 $T(n) = 9T(\frac{n}{3}) + n$

$\log_3 9 = 2 > k = 1$

$\theta(n^2)$ case 1

<u>Ex5</u> $T(n) = 2T(\frac{n}{2}) + n$

$\log_2 2 = 1 = k = 1, \; p = 0$

case 2: $\Theta(n \log n)$

<u>Ex6</u> $T(n) = 4T(\frac{n}{2}) + n^2 \log^5 n$

$\log_2 4 = 2, \; k = 2$

when $\log_b a = k$, and $p > -1$, multiply by $\log n$

case 2: $\Theta(n^2 \log^6 n)$

<u>Ex7</u> $T(n) = 2T(\frac{n}{2}) + \frac{n}{\log n}$

$\log_2 2 = 1, \; k = 1, \; p = -1$

case 2: $\Theta(n \log \log n)$ – $\log\log n$ since $p = -1$

<u>Ex8</u> $T(n) = 2T(\frac{n}{2}) + \frac{n}{\log^2 n}$

$\log_2 2 = 1, \; k = 1, \; p = -2$

case 2: $\Theta(n)$ – ignore log since $p < -1$

<u>Ex9</u> $T(n) = T(\frac{n}{2}) + n^2$

$\log_2 1 = 0 < k = 2, \; p = 0$

case 3: $\Theta(n^2)$

<u>Ex10</u> $T(n) = 2T(\frac{n}{2}) + n^2 \log^2 n$ $\quad p = 2$

$\log_2 2 = 1 < k = 2$

copy expression since $\log_b a < k$

case 3: $\Theta(n^2 \log^2 n)$

<u>Ex11</u> $T(n) = 4T(\frac{n}{2}) + \frac{n^3}{\log n}$

$\log_2 4 = 2 < k = 3, \; p = -1$

case 3: $\Theta(n^3)$ – ignore log in denominator since $p < 0$

Jacky Ly

15

2.4.2 Master Theorem Examples

Case 1:

$T(n) = 2T(\frac{n}{2}) + 1 - \Theta(n^1)$ $\quad \log_2 2 = 1 > k = 0$

$T(n) = 4T(\frac{n}{2}) + 1 - \Theta(n^2)$ $\quad \log_2 4 = 2 > k = 0$

$T(n) = 4T(\frac{n}{2}) + n - \Theta(n^2)$ $\quad \log_2 4 = 2 > k = 1$

$T(n) = 8T(\frac{n}{2}) + n^2 - \Theta(n^3)$ $\quad \log_2 8 = 3 > k = 2$

$T(n) = 16T(\frac{n}{2}) + n^2 - \Theta(n^4)$ $\quad \log_2 16 = 4 > k = 2$

Case 3:

$T(n) = T(\frac{n}{2}) + n - \Theta(n)$ $\quad \log_2 1 = 0 < k = 1$

$T(n) = 2T(\frac{n}{2}) + n^2 - \Theta(n^2)$ $\quad \log_2 2 = 1 < k = 2$

$T(n) = 2T(\frac{n}{2}) + n^2 \log n - \Theta(n^2 \log n)$ $\quad \log_2 2 = 1 < k = 2,\ p = 1$

$T(n) = 4T(\frac{n}{2}) + n^3 \log^2 n - \Theta(n^3 \log^2 n)$ $\quad \log_2 4 = 2 < k = 3,\ p = 2$

$T(n) = 2T(\frac{n}{2}) + \frac{n^2}{\log n} - \Theta(n^2)$ $\quad \log_2 2 = 1 < k = 2,\ p = -1$

Case 2:

$T(n) = T(\frac{n}{2}) + 1 - \Theta(\log n)$ $\quad \log_2 1 = 0 = k = 0$

$T(n) = 2T(\frac{n}{2}) + n - \Theta(n \log n)$ $\quad \log_2 2 = 1 = k = 1$

$T(n) = 2T(\frac{n}{2}) + n \log n - \Theta(n \log^2 n)$ $\quad \log_2 2 = 1 = k = 1,\ p = 1$

$T(n) = 4T(\frac{n}{2}) + n^2 - \Theta(n^2 \log n)$ $\quad \log_2 4 = 2 = k = 2$

$T(n) = 4T(\frac{n}{2}) + (n \log n)^2 - \Theta(n^2 \log^3 n)$ $\quad \log_2 4 = 2 = k = 2,\ p = 2$

$T(n) = 2T(\frac{n}{2}) + \frac{n}{\log n} - \Theta(n \log \log n)$ $\quad \log_2 2 = 1 = k = 1,\ p = -1$

$T(n) = 2T(\frac{n}{2}) + \frac{n}{\log^2 n} - \Theta(n)$ $\quad \log_2 2 = 1 = k = 1,\ p = -2$

2.5 Recurrence Relation (Root function)

$$T(n) = \begin{cases} 1 & , n = 2 \\ T(\sqrt{n}) + 1 & , n > 2 \end{cases}$$

$T(n) = T(n^{1/2}) + 1$ ①

$T(n) = T(n^{1/2^2}) + 2$ ②

$T(n) = T(n^{1/2^3}) + 3$ ③

$\vdots$

$T(n) = T(n^{1/2^k}) + k$ ④

Assume $n = 2^m$

$T(2^m) = T(2^{m/2^k}) + k$

Assume $T(2^{m/2^k}) = T(2)$

$\therefore \frac{m}{2^k} = 1,\ m = 2^k$ and $k = \log_2 m$

$\because n = 2^m,\ m = \log_2 n$

So, $k = \log \log_2 n$

$\Theta(\log \log_2 n)$

objective:
$\max \sum x_i p_i$

3.1 Knapsack Problem - Greedy Method

constraint: $\sum x_i w_i \le m$

$n = 7$
$m = 15$
$0 \le x \le 1$

object (o):	1	2	3	4	5	6	7
profit (p):	10	5	15	7	6	18	3
weight (w):	2	3	5	7	1	4	1
p/w:	5	1.3	3	1	6	4.5	3
x	(1	2/3	1	0	1	1	1)
	x_1	x_2	x_3	x_4	x_5	x_6	x_7

weight:
$\sum x_i w_i = 1\times 2 + \frac{2}{3}\times 3 + 1\times 5 + 0\times 7 + 1\times 1 + 1\times 4 + 1\times 1$
$2 + 2 + 5 + 0 + 1 + 4 + 1 = 15$

profit:
$\sum x_i p_i \quad 1\times 10 + \frac{2}{3}\times 5 + 1\times 15 + 0\times 7 + 1\times 6 + 1 + 18 + 1\times 3$
$= 10 + 1.3 + 15 + 6 + 18 + 3 = 54.6$

3.2 Job Sequencing with Deadlines - Greedy Method

Ex1 $n = 5$

Jobs	J_1	J_2	J_3	J_4	J_5
Profits	20	15	10	5	1
Deadlines	2	2	1	3	3
	✓	✓	×	✓	×

0 $\underline{J_2}$ 1 $\underline{J_1}$ 2 $\underline{J_4}$ 3

$\{J_2, J_1, J_4\}$
$J_2 \to J_1 \to J_4$
$J_1 \to J_2 \to J_4$

Total profit: $15 + 20 + 5 = 40$

Ex2 $n=7$

Jobs	J_1	J_2	J_3	J_4	J_5	J_6	J_7
Profits	35	30	25	20	15	10	5
Deadlines	3	4	4	2	3	1	2

0 $\underline{J_4}$ 1 $\underline{J_3}$ 2 $\underline{J_1}$ 3 $\underline{J_2}$ 4

Total profit: $20 + 25 + 35 + 30 = 110$

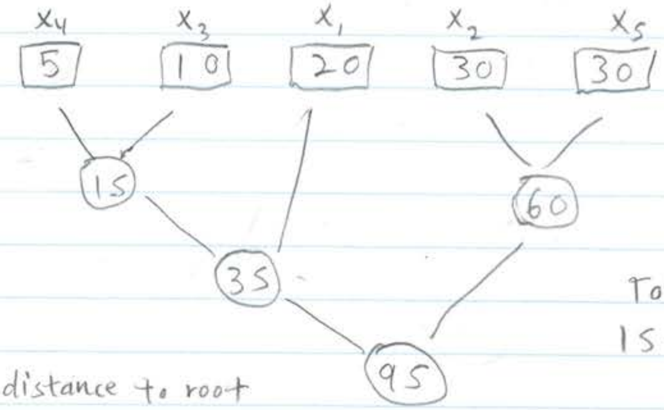
(17)

Jacky Ly

3.3 Optimal Merge Pattern - Greedy Method

lists → x_1 x_2 x_3 x_4 x_5

sizes → 20 30 10 5 30



* select minimum/ lowest values to merge first

Total cost:
$15+35+60+95 = 205$

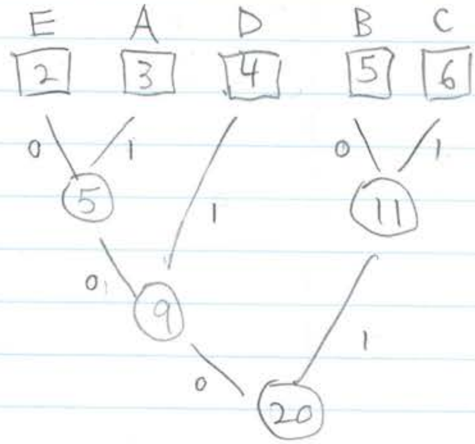
distance to root
↓
$\sum d_i x_i$ - total cost formula

alternative method (count tree level to root):
$3\times\boxed{5} + 3\times\boxed{10} + 2\times\boxed{20} + 2\times\boxed{30} + 2\times\boxed{30} = 205$

3.4 Huffman Coding - Greedy Method

Message: BCCABBDDAECCBBAEDDCC

* left side - 0
right side - 1



char	count	code	# of bits
A	3	001	$3\times3=9$
B	5	10	$5\times2=10$
C	6	11	$6\times2=12$
D	4	01	$4\times2=8$
E	2	000	$2\times3=6$
5×8 bits $=40$	20	12 bits	45 bits

8 bits for ASCII

message size = 45 bits
table size = 40 + 12 = 52 bits
total size = 45 + 52 = 97 bits

* follow path from root to find code

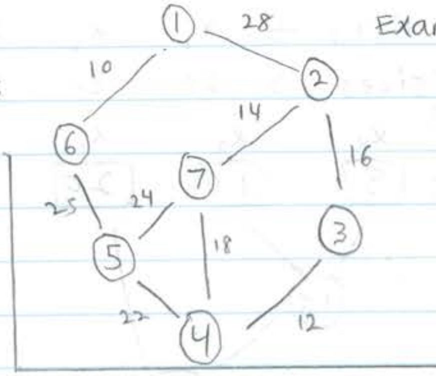
Decoding: start from root to find character
B = 10, C = 11, etc.

$\sum d_i \cdot f_i$ (total bits)
$= 3\times2 + 3\times3 + 2\times4 + 2\times5 + 2\times6$
$= 45$ bits

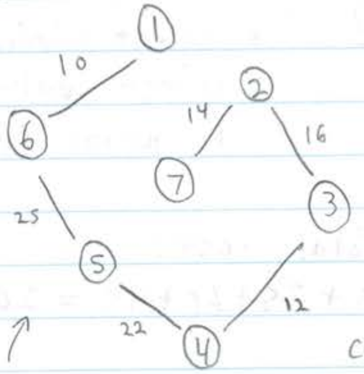
3.5 Prim's and Kruskal's Algorithms - Greedy Method

Prim's: select minimum weights but must be connected to newly selected vertices

Example:



$10 \to 25 \to 22 \to 12 \to 16 \to 14$

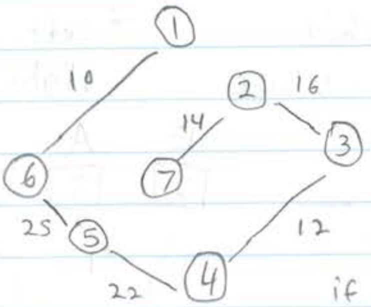


cost: 99

minimum spanning tree

Note: minimum cost spanning tree can't be obtained for non connected graphs!
(no algorithm can be used)

Kruskal's: always select minimum cost edge, but must not form a cycle



$10 \to 12 \to 14 \to 16 \to 22 \to 25$

cost: 99

$|E|$ - num of edges

$|V| - 1$

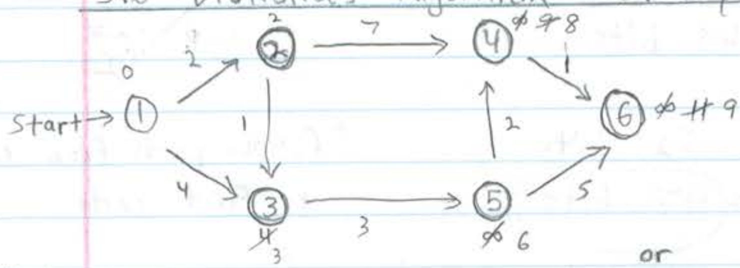
if using min heap: $\theta(n \log n)$

$\theta(|V||E|)$

$\theta(n \cdot e) = \theta(n^2)$

*only one optimal cost

3.6 Dijkstra's Algorithm - Greedy Method



Relaxation:

if $(d[u] + c(u,v) < d[v])$
$d[v] = d[u] + c(u,v)$

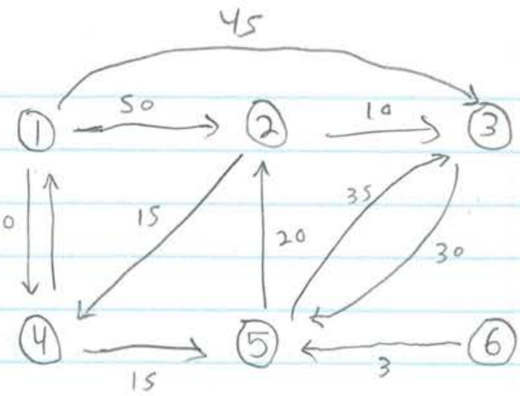
v	d[v]
2	2
3	3
4	8
5	6
6	9

time complexity: $n = |V|$ $|V| = n \cdot n = n^2$ $\theta(|V|^2)$ or $\theta(n^2)$

$1 \xrightarrow{2} 2 \xrightarrow{3} 3 \xrightarrow{6} 5 \xrightarrow{8} 4 \xrightarrow{9} 6$

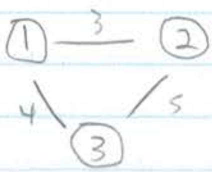
Jacky Ly

Ex starting vertex 1

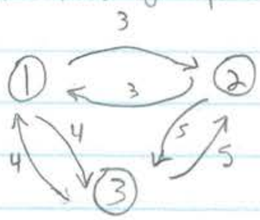


selected vertex	2	3	4	5	6
4	50	45	10	∞	∞
5	50	45	10	25	∞
2	45	45	10	25	∞
3	45	45	10	25	∞ ← no change
6	45	45	10	25	∞ ← can't reach 6

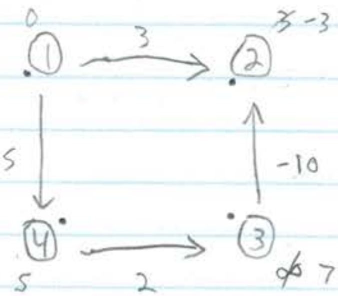
can also work on non directed graphs!



=



Drawbacks:



*negative weights/edge

correct answer is -3,
but algorithm returned 3