no need to write psvedo code

Final review

1) floy'd algorithm
2) knapsack problem (DP or Greedy)
3) time complexity for a given psvedocode (iterative)
4) two problems : Asymptotic order
5) time complexity for a given psvedocode (recursive)
6) Dijkstra algorithm (true or false)
7) P=NP Give you a problem we covered in class
8) N- coloring problem
9) travelling salesperson problem tsp - decision , tsp search , tsp - opt
   - Tsp = Backtracking
   - Tsp = Branch & Bound
   - Tsp = Dp

$2+\left(\frac{n}{4}\right)+\log_2 n \in O(2^n)$

use master theorem

```
for(k=0, k<n; k++)  -> n
    for (p=1, p<(n-3), p++)  -> n
        for (j=2; j<p; j=j^2) -> Log Log n
            print(...)
```

not recursive
then drop : master theorem
Back substitution

$O(n^2 \log\log n)$

Why its $\log\log n$?

j	2^k	2^{2^k}
2	2^1	2^{2^0}
4	2^2	2^{2^1}
16	2^4	2^{2^2}
256	2^8	2^{2^3}
n		2^{2^k}

$n = 2^{2^k}$

$\log_2 n = 2k$

$\log\log n = k$

```
function f(n) {
    If n≤1  return 1   X
    for (i=0, i<(n-3), i++)  -> n
        for (j=n; j>1, j=j/2)  -> Log n -> 2^k = n/(2^k)
            Print(---)
```

master theorem

$f(\frac{n}{2})$ $f(\frac{n}{2})$ $f(\frac{n}{2})$ $f(\frac{n}{2})$ $\Big\rangle$ $4T(\frac{n}{2})$

$\Rightarrow 4T(\frac{n}{2}) + n\log n$

$a=4$ $p=1$

$b=2$

$k=1$

(H) $(n^{\log_b a})$

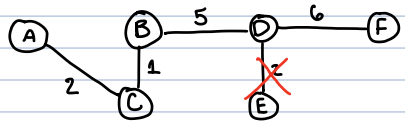
(H) (n^2)

$\log_2 4 = 2 > k$

{ V, W }

Visited	B	C	D	E	F	
{A}	A,4	A,2	A,∞	A,∞	A,∞	
{A,C}	C,3	X	C,10	C,12	C,∞	
{A,C,B}	X	X	B,8	C,12	B,∞	
{A,C,B,D}	X	X	X	D,10	D,14	
{A,C,B,D,E}	X	X	X	X	D,14	

Solution:



Cost(A, V_i) = min({A, V_i}, {A, {V_k}, V_i})

{ V, W }

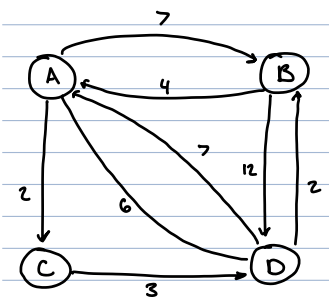
Visited	B	C	D	E	F	
{A}	A,4	A,2	A,∞	A,∞	A,∞	

A	B	C	D	E	f
0	1	2	3	4	5
∅	(0,4)	(0,2)	(0,∞)	(0,∞)	(0,∞)

Visited

0	1	2	3	4	5
1	0	1	0	0	0

P	A	B	C	D	E	F
A	0					
B		0				
C			0			
D				0		
E					0	
F						0



D	A	B	C	D
A	∞	7	2	6
B	4	∞	∞	12
C	∞	∞	∞	3
D	7	1	∞	∞

P	A	B	C	D
A	∞	0	0	0
B	0	∞	0	0
C	0	0	∞	0
D	0	0	0	∞

P	A	B	C	D
A	∞	D	D	C
B	0	∞	A	C
C	D	D	∞	0
D	B	0	B	∞

Floyd-Marshall algorithm

use D^0

(A)

D^0	A	B	C	D
A	∞	7	2	6
B	4	∞	6	10
C	∞	∞	∞	3
D	7	1	9	∞

$Cost(B,C) = \min\{B\to C, B\to A\to C\} = 6$
∞ , $4+2=6$

$Cost(B,D) = \min\{B\to D, B\to A\to D\} = 10$
12 , $4+6=10$

$Cost(C,B) = \min\{C\to B, C\to A\to B\} = x$
∞ , ∞

$Cost(C,D) = \min\{C\to D, C\to A\to D\}$
3, ∞

$Cost(D,B) = \min\{D\to B, D\to A\to B\} =$
1,

use D^0

(B)

D^1	A	B	C	D
A	∞	7	2	6
B	4	∞	6	10
C	∞	∞	∞	3
D	5	1	7	∞

$\min\{A\to C, A\to B\to C\}$
2 , 7...

$\min\{A\to D, A\to B\to D\}$
6 , 7...

$\min\{C\to A, C\to B\to A\}$
∞ ∞

$\min\{C\to D, C\to B\to D\}$
3 , ∞..

$\min\{D\to A, D\to B\to A\} = 5$
7, $1+4=5$

$\min\{D\to C, D\to B\to C\} = 7$
9 , $1+6=7$

(C) use D^1

D^2	A	B	C	D
A	∞	7	2	5
B	4	∞	6	9
C	∞	∞	∞	3
D	5	1	7	∞

$\min\{A\to B, A\to C\to B\}$
7, $2+\infty$

$\min\{A\to D, A\to C\to D\} = 5$
6 , $2+3=5$

$\min\{B\to A, B\to C\to A\}$
4, 6...

$\min\{B\to D, B\to C\to D\} = 9$
10 , $6+3=9$

$\min\{A\to B, A\to C\to B\}$

(D) → j

i ↓	3	0	1	2	3
	D	A	B	C	D
0	A	∞	6	2	5
1	B	4	∞	6	9
2	C	8	4	∞	3
3	D	5	1	7	∞

* General formula: $M_k[i][j] = \min\{M_{k-1}[i][j], M_{k-1}[i][k] + M_{k-1}[k][j]\}$

$M_3[0][1] = \min\{M_2[0][1], M_2[0][3] + M_2[3][1]\} = 6$
7 , $5+1=6$

A, C
2, 2

B, A 4, ∞
B, C 6, 9
C, A ∞, $3+5=8$
C, B ∞, $3+1=4$

P	A	B	C	D
A	∞	D	D	C
B	D	∞	A	C
C	D	D	∞	D
D	B	D	B	∞

Min(A, D) = A → D → B
(1)

A → C → D → B
(2)
)

2 + 3 + 1 = 6

Min(D, A) = D → B → A
(1)
0

= D → B → A
0

= D → B → A

Pseudo Code:

```
function MinPath(D , P):
    for(k=0; k<n; k++)
        Create matrix M & initialize it to -1
        for(i=0; i<n; i++)
            for(j=0; j<n; j++)
                if i=k & j=k then M[i][j] = D[i][j]
                else
                    M[i][j] = min( D[i][j], D[i][k] + D[k][j]
                    if M[i][j] < D[i][j] then P[i][j] = k
    D=M
return P;
```

Bellman-Ford

HW #1

$$\sum_{i=10}^{n} \sum_{j=1}^{\log_2 n + 1} j \qquad \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=10}^{n} \frac{\log_2 n + 1 (\log_2 n + 2)}{2}$$

$$\frac{1}{2} \sum_{i=10}^{n} (\log_2 n)^2 + 2\log_2 n + \log_2 n + 2$$

$$\frac{1}{2} \left[\sum_{i=10}^{n} (\log_2 n)^2 + 3 \sum_{i=10}^{n} \log_2 n + \sum_{i=10}^{n} 2 \right]$$

$$\frac{1}{2} \left[\left(\sum_{i=1}^{n} (\log_2 n)^2 - \sum_{i=1}^{9} (\log_2 n)^2 \right) + \left(3 \sum_{i=1}^{n} \log_2 n - 3 \sum_{i=1}^{9} \log_2 n \right) + \left(\sum_{i=1}^{n} 2 - \sum_{i=1}^{9} 2 \right) \right]$$

$$\frac{1}{2} (n\log_2^2 n - 9\log_2^2 n + 3n\log_2 n - 27\log_2 n + 2n - 18)$$

```
For (i = 1; i <= (n - 2); i++)
    for (j = i + 1; j <= (n - 1); j++)
        for (k = + 1; k <= n; k++)
            print(...)
```

i	j	k
1	2, 3, 4, ..., n-1	3, 4, 5, ... n
2	3, 4, 5, ... (n-1)	
n-2	n-1	n

$$T(n) = \sum_{i=1}^{n-2} \sum_{j=i+1}^{n-i-1} \sum_{k=1}^{n-j} 1$$

$n^2 - in - in + i^2 - n + i$

$$= \sum_{i=1}^{n-2} \sum_{j=1}^{n-i-2} n-j = \sum_{i=1}^{n-2} \sum_{j=1}^{n-i-1} n - \sum_{i=1}^{n-2} \sum_{j=1}^{n-i-1} j = \sum_{i=1}^{n-2} n^2 - in - n - \sum_{i=1}^{n-2} \frac{(n-i-1)(n-i)}{2}$$

$$= \sum_{i=1}^{n-2} n^2 - n \sum_{i=1}^{n-2} i - \sum_{i=1}^{n-2} n - \frac{1}{2} \left(\sum_{i=1}^{n-2} n^2 - 2n \sum_{i=1}^{n-2} i + \sum_{i=1}^{n-2} i^2 - \sum_{i=1}^{n-2} n + \sum_{i=1}^{n-2} i \right)$$

$$= (n-2)n^2 - n\left(\frac{n-2(n-1)}{2}\right) - (n-2)n - \frac{1}{2}\left((n-2)n^2 - 2n\left(\frac{n-2(n-1)}{2}\right) + \frac{(n-2)(n-1)(2n-3)}{2} - (n-2)n + \frac{(n-4)(n-1)}{2}\right.$$

$O(n^3)$

4.

$P_{out} = \frac{1}{4}$ $\quad P_{in}\left(P_n\left(\frac{1}{4}\right)\left(\frac{3}{8}\right) + P_{n-1}\left(\frac{1}{5}\right)\left(\frac{3}{4}\right) + P_{n-2}\ ?\ \right) = \frac{3}{4}$

$\frac{3}{32}$ $\qquad$ $\frac{3}{20}$

$+ x = \frac{3}{4} \rightarrow x = \frac{3}{4} - \frac{3}{32} - \frac{3}{20} \rightarrow P_{n-2} = \frac{81}{60} \rightarrow \frac{81}{160(4-2)}$

$$A(n) = \frac{n}{4} + \frac{3n}{32} + \frac{3(n-1)}{20} + \frac{81}{160(n-2)} \sum_{i=1}^{n-2}$$

$$= \frac{8n + 3n + 5/8(n-1)}{32} + \frac{81}{160(n-2)} \cdot \frac{(n-2)(n-1)}{2}$$

$$\hookrightarrow \frac{8n + 3n + 5/8(n-1)}{32} + \frac{81}{160}$$

HW2

$$T(n) = 9T\left(\frac{n}{3}\right) + n^3 + n^2 + n + 1$$

$$T\left(\frac{n}{3}\right) = 9\left[9T\left(\frac{n}{3^2}\right) + \left(\frac{n}{3}\right)^3 + \left(\frac{n}{3}\right)^2 + \left(\frac{n}{3}\right) + 1\right] + n^3 + n^2 + 1$$

$$= 9^2 T\left(\frac{n}{3^2}\right) + \frac{n^3}{3} + n^2 + 3n + 9 + n^3 + n^2 + 1$$

$$T\left(\frac{n}{3^2}\right) = 9\left[9^2\left(\frac{n}{3^3}\right) + \frac{n^3}{3^6} + \frac{n^2}{3^4} + \frac{n}{3^2} + 1\right] + \frac{n^3}{3} + n^2 + 3n + 9 + n^3 + n^2 + 1$$

$$= 9^3\left(\frac{n}{3^3}\right) + \frac{n^3}{3^2} + n^2 + 3^2 n + 9^2 + \frac{n^3}{3} \quad \cdots$$

k steps

$T(n)\ 9T(\frac{n}{3}) + n^3 + n^2 + n + 1$

$T(n) = 9^{\log_3 n}(1)T\ \frac{3n^3}{2}\left(\frac{1}{3^{\log_3 n}} - 1\right) + n\log_3 n + n\left(\frac{3^{\log_3 n} - 1}{2}\right) + \frac{9^{\log_3 n} - 1}{8}$

$t(n) = n^2 - \frac{3n^3}{2}\left(\frac{1}{n} - 1\right) + n\log_3 n + n\left(\frac{n-1}{2}\right)$

$\theta(n^3)$

HW # 2 pseudo code

```
function SortFile ( file-path , size-file , size buffer )
    I1 : of size   size buffer
    I2 :
    O1 :
    file = Load-file (file path)
    number of passes = Log2 (size of file) - 1
    number of files = Size of file / size of buffers
    first-pass (file , number of files)
    next-pass (1, number of passes , number files , size of file)
```

```
function FirstPass (file , number of files)
    While file is not empty {
      I1 = read(size of buffer)
      O1 = Quicksort (I1)
      O1 = flush ( tmpfile
```

```
function nextPass (currentpass , number-passes , number of files, size of file )
    If currentpass = number of pasres then return
    file Index = 0
    While true :
        file 1 = Load (currentpass, file Index - 1 )
        file 2 = Load (currentpass, file Index )
        While file 1 and file 2 are not empty
          I1  read (file 1)
          I2  read (file 2)
          O1  Merge (I1, I2)
        file Index = file Index + 2
        If file Index ≥ number of files  break
    nextpass (currentpass + 1 , number of passes, number files / 2 , size file * 2 )
```

$t(n) = T\left(\frac{n}{\frac{M}{B}}\right) + \frac{N}{B}\log_{\frac{M}{B}}\left(\frac{N}{B}\right)$

$t(n) = T\left(\frac{N}{2}\right) + \frac{N}{4}\log_2 \frac{N}{4}$

master theorem

$a = 1 \quad b = 2 \quad k = 1 \quad p = 1$

$\log_2 1 = 0 < k$

case #3 → 1

$p = 1$

$\theta(n^k \log^p n)$

$\theta\left(\frac{N}{B}\log_{\frac{M}{B}}\left(\frac{N}{B}\right)\right)$

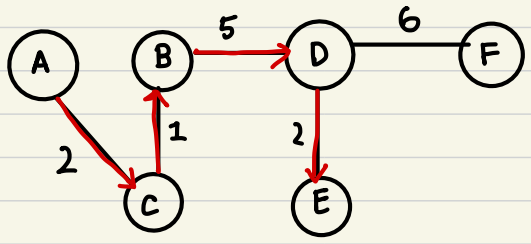
Dijkstra Algorithm (greedy)

10/30/2023

Find the minimum path from the source node to all the other nodes

{V,W}	B	C	D	E	F
A	A,4	A,2	A,∞	A,∞	A,∞
A,C	C,3	X	C,10	C,12	C,∞
A,C,B	X	X	B,8	C,12	B,∞
A,C,B,D	X	X	X	D,10	D,14
A,C,B,D,E	X	X	X	X	D,14

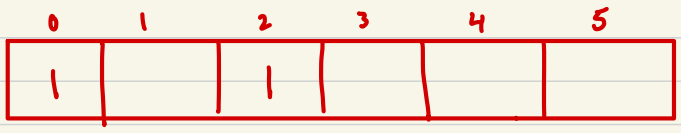
Solution



Destination	path	cost
B	{A,C,B}	3
C	{A,C}	2
D	{A,C,B,D}	8
→ E	{A,C,B,D,E}	10
F	{A,C,B,D,F}	14

$cost(A, V_i) = Min(\{A, V_i\}, \{A, \{V_k\}, V_i\})$

Visited



space complexity

$t(n) = n + n^2$

$= O(n^2)$

time complexity

$O(n^2)$

Min {A,B} , {A,C,B}

4 , 2 + 1 = 3

= 3

Min {A,D} , {A,C,D}

∞ , 2 + 8 = 10

= 10

Min {A,E} , {A,C,E}

∞ , 2 + 10 = 12

= 12

Min {A,F} , {A,C,F}

∞ , 2 +

min {A,C,D} , {A,C,B,D}

10 , 2+1+5 = 8

= 8

{A,C,E} , {A,C,B,E}

12 , 2 + 1 + ∞

= 12

{A,C,F} , {A,C,B,F}

∞ , 2+1+∞

{A,C,B,E} , {A,C,B,D,E}

12 , 2+1+5+2 = 10

= 10

{A,C,B,F} , {A,C,B,D,F}

∞ , 2+1+5+6 = 14

= 14

{A,C,B,D,F} , {A,C,B,D,E,F}

14 , 2+1+5+2+5 = 15

= 14

Visited	B	C	D	E	F
{A}	A,4	A,2	A,∞	A,∞	A,∞

0	1	2	3	4	5
A	B	C	D	E	F
0	(0,4) ↓ (2,3)	(0,2)	(0,∞)	(0,∞)	(0,∞)

Visited =	A 0	B 1	C 2	D 3	E 4	F 5
	1	0	1	0	0	0

$t(n) = n + n = 2n$

for each V in graph : → N

Compute Dijkstra → N^2

$O(n^3)$

Encryption algorithmn

	Data	codes
user 1	01	0000
user 2	10	1111
user 3	11	

bits → volts

$f(b) = v$

two rules

(1) $f(0) = +1v$

(2) $f(1) = -1v$

use XOR operation

1 XOR 1 = 0

0 XOR 0 = 0

1 XOR 0 = 1

0 XOR 1 = 1

Receiver

$f(b) = v$

$$f\left(\frac{\sum_{i=1}^{c} f_i C_i}{\text{Length(code)}}\right) = \text{bit}$$

+1v, +1v, +1v, +1v

−1v, −1v, −1v, −1v

−1v, −1v, −1v, −1v

$= -4v$

	Data	codes
user 1	01	0000
user 2	10	1010
user 3	11	1100

+1v, +1v, +1v, +1v

−1v, −1v, +1v, +1v

(−1v) + (−1v) + 1v + 1v

$= 0$

	Data	codes
user 1	01	0000
user 2	10	1010
user 3	11	1100

V1 = 00001111

XOR 00000000

00001111

V2 11110000

XOR 10101010

01011010

V3 1111 1111

XOR 1100 1100

00110011

V1	0	0	0	0	1	1	1	1
+1v								
−1v								
V2	0	1	0	1	1	0	1	0
+1v								
−1v								
V3	0	0	1	1	0	0	1	1
+1v								
−1v								
$\sum f_i$	+3v	+1v	+1v	−1v	−1v	+1v	−3v	−1v

+3v, +1v, +1v, −1v, −1v, +1v, −3v, −1v

−1v, +1v, −3v, −1v, −1v, +1v, −1v, +1v

$(-3v) + (1v) + (-1v) + (-1v) = \frac{-4v}{4} = -1v$

$(1v) + (1v) + (3v) - 1v = \frac{4v}{4} = 1v$

f(−1v) = 1 bit

f(+1) = 0 bit

Decode $\left(\frac{\sum_{i=1}^{\text{Len(code)}} f_i C_i}{\text{Length(code)}}\right) = \text{bit}$

$f(1\text{ bit}) = -1v$

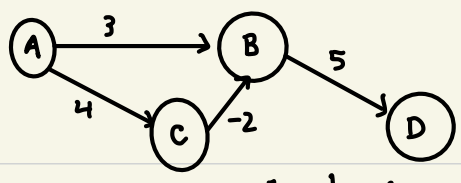
$f(0\text{ bit}) = +1v$

$f^{-1}(-1v) = 1\text{ bit}$

$f^{-1}(+1v) = 0\text{ bit}$

Psvedo code

next page



Bellman - ford (Distance vector version)

P = NP

from

	A	B	C	D
A	0	3	4	∞
B	∞	0	∞	∞
C	∞	∞	0	∞
D	∞	∞	∞	0

to

	A	B	C	D
A	0	∞	∞	∞
B	∞	0	∞	∞
C	∞	-2	0	∞
D	∞	∞	∞	0

	A	B	C	D
A	0	∞	∞	∞
B	∞	0	∞	5
C	∞	∞	0	∞
D	∞	∞	∞	0

	A	B	C	D
A	0	∞	∞	∞
B	∞	0	∞	∞
C	∞	∞	0	∞
D	∞	∞	∞	0

Step 2

	A	B	C	D
A	0	2	4	8
B	∞	0	∞	5
C	∞	-2	0	3
D	∞	∞	∞	0

$Dist_A(B) = \min \begin{cases} A \to B + B \to B, & 3+0 = 3 \\ A \to C + C \to B, & 4+(-2) = 2 \\ A \to D + D \to B, & \infty + \dots \end{cases}$

$= 2$

$Dist_A(C) = \min \begin{cases} A \to C + C \to C = 4+0 \\ A \to B + B \to C = 5+-3 \\ A \to D + D \to C = \infty + x \end{cases}$

$= 4$

$Dist_A(D) = \min \begin{cases} cost(A,D) + Dist_D(D), & \infty + 0 \\ cost(A \to B) + Dist_B(D), & 3+5 = 8 \\ cost(A \to C) + Dist_C(D), & 4+\infty \end{cases}$

$= 8$

$Dist_B(A) = Min \begin{cases} Cost(B,A) + Dist_A(A), & \infty + 0 \\ Cost(B,C) + Dist_C(A), & \infty + \infty \\ Cost(B,D) + Dist_D(A), & 5 + \infty \end{cases}$

$= \infty$

$Dist_B(C) = Min \begin{cases} Cost(B,C) + D, & \infty \\ Cost(B,A) + Dist_A(C), & \infty \\ Cost(B,D) + Dist_D(C), & 5 \end{cases}$

Matrix(n) = Matrix(n)-1

	A	B	C	D
A	0			
B		0		
C			0	
D				0

time complexity

Step 1

Step 2 matrix → matrix → .... + n-1

so a for loop

Traveling Sales person Branch & Bound

Here is the reduced matrixes

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	∞	∞	11	2	0
3	0	∞	∞	0	2
4	15	∞	12	∞	0
5	11	∞	0	12	∞

from 1-2

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	1	∞	∞	2	0
3	∞	3	∞	0	2
4	4	3	∞	∞	0
5	0	0	∞	12	∞

from 1 -3

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	12	∞	11	∞	0
3	0	3	∞	∞	2
4	∞	3	12	∞	0
5	11	0	0	∞	∞

from 1-4

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	10	∞	9	0	∞
3	0	3	∞	0	∞
4	12	0	9	∞	∞
5	∞	0	0	12	∞

from 1-5

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	∞	∞	11	∞	0
3	0	∞	∞	∞	2
4	∞	∞	∞	∞	∞
5	11	∞	0	∞	∞

From 1 - 4 - 2

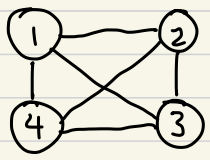
	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	1	∞	∞	∞	0
3	∞	1	∞	∞	0
4	∞	∞	∞	∞	∞
5	0	0	∞	∞	∞

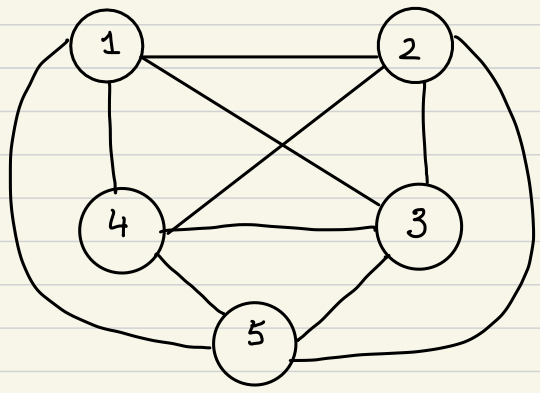
from 1-4-3

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	1	∞	0	∞	∞
3	0	3	∞	∞	∞
4	∞	∞	∞	∞	∞
5	∞	0	0	∞	∞

from 1-4-5

Traveling Salesman problem Branch & Bound





Step 1: reduce matrix

	1	2	3	4	5	min
1	∞	20	30	10	11	10
2	15	∞	16	4	2	2
3	3	5	∞	2	4	2
4	19	6	18	∞	3	3
5	16	4	7	16	∞	4
						21

Subtract min in matrix →

	1	2	3	4	5
1	∞	10	20	0	1
2	13	∞	14	2	0
3	1	3	∞	0	2
4	16	3	15	∞	0
5	12	0	3	12	∞
min	1	0	3	0	0

Subtract from columns

	1	2	3	4	5
1	∞	10	17	0	1
2	12	∞	11	2	0
3	0	3	∞	0	2
4	15	3	12	∞	0
5	11	0	0	12	∞

21 + 4 = 25

	1	2	3	4	5
1	∞	20	30	10	11
2	15	∞	16	4	2
3	3	5	∞	2	4
4	19	6	18	∞	3
5	16	4	7	16	∞

Original matrix problem

	1	2	3	4	5
1	∞	10	17	0	1
2	12	∞	11	2	0
3	0	3	∞	0	2
4	15	3	12	∞	0
5	11	0	0	12	∞

reduce Cost = 25

we use reduce matrix now that we solved it

from 1 → 2

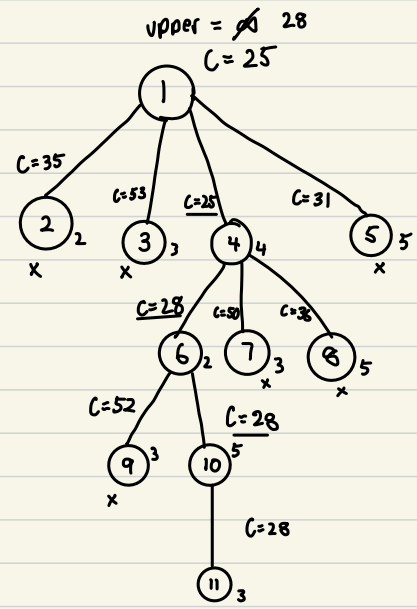
	1	2	3	4	5	min
1	∞	∞	∞	∞	∞	∞
2	∞	∞	11	2	0	0
3	0	∞	∞	0	2	0
4	15	∞	12	∞	0	0
5	11	∞	0	12	∞	0
min	0	∞	0	0	0	

from reduced matrix(1,2) → Cost(1,2) + reduced cost + reduction

10 + 25 + 0 = 35

From 1-3

	1	2	3	4	5	min
1	∞	∞	∞	∞	∞	∞
2	12	∞	∞	2	0	0
3	∞	3	∞	0	2	0
4	15	3	∞	∞	0	0
5	11	0	∞	12	∞	0
Min	11	0	∞	0	0	



reduce column by 11

From 1-3

	1	2	3	4	5	min
1	∞	∞	∞	∞	∞	∞
2	1	∞	∞	2	0	0
3	∞	3	∞	0	2	0
4	4	3	∞	∞	0	0
5	0	0	∞	12	∞	0
Min	11	0	∞	0	0	

Cost(1,3) + reduced cost + reduction

17 + 25 + 11 = 53

From 1-4-2

	1	2	3	4	5	min
1	∞	∞	∞	∞	∞	∞
2	∞	∞	11	∞	0	0
3	0	∞	∞	∞	2	0
4	∞	∞	∞	∞	∞	∞
5	11	∞	0	∞	∞	0
Min	0	∞	0	∞	0	

Cost(4,2) + Cost(4) + reduction

3 + 25 + 0 = 28

from 1-4-2-3

	1	2	3	4	5	min
1	∞	∞	∞	∞	∞	
2	∞	∞	∞	∞	∞	
3	∞	∞	∞	∞	0	2
4	∞	∞	∞	∞	∞	
5	0	∞	∞	∞	∞	11
Min	0	∞	∞	∞	0	

Cost(2,3) + Cost(6) + reduction

11 + 28 + 13 = 52

Shortest nodes

1-4-2-5-3-1

Knapsack problem greedy method

object	1	2	3	4	5	6	7
profit	10	5	15	7	6	18	3
weight	2	3	5	7	1	4	1
$\frac{p}{w}$	5	1.3	3	1	6	4.5	3
	1	2/3	1	X	1	1	1
$0 \le x \le 1$	x_1	x_2	x_3	x_4	x_5	x_6	x_7

Constraint

$\sum x_i w_i \le$ Capacity of Bag

Objective

$\max \sum x_i p_i$

15kg - x_5 weight = 14 — 15 − 1 = 14

14 - x_1 weight = 12 — 14 − 2 = 12

12 - x_6 weight = 8 — 12 − 4 = 8

8 - x_3 weight = 3 — 8 − 5 = 3

3 - x_7 weight = 2 — 3 − 1 = 2

2 - 2/3 of x_2 weight = 0 — 2 − 2 = 0

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

$$\sum x_i p_i = 1\times 10 + \tfrac{2}{3}\times 5 + 1\times 15 + 1\times 6 + 1\times 18 + 1\times 3$$
$$10 + 2\times 1.3 + 15 + 6 + 18 + 3 = 54.6$$

problem 2 knapsack

W=10

i	Wi	Pi	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	20	40	0	0	0	0	0	0	0	0	0	0	0
2	6	24	0	0	0	0	0	0	24	24	24	24	24
3	3	6	0	0	0	6	6	6	24	24	24	30	30
4	7	40	0	0	0	6	6	6	24	40	40	40	(46)

$$M[i][w] = \max\left(\underbrace{M[i-1][w]}_{24},\ \underbrace{M[i-1][w-w_i] + P[i]}_{6}\right)$$

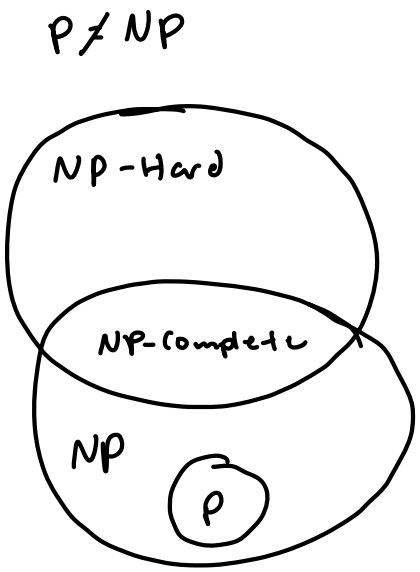
$\{\emptyset\} = 0$

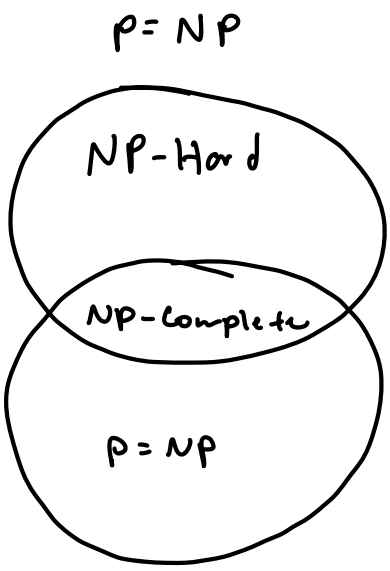
$\{1\} = 0$

$\{2\} = 1$

$\{3\}$

$\{\#4, \#3, \emptyset\}$





Decidable

↓

Decision

Alan Turing

P = the set of decidable problems that can be solved in polynomial time w/ a deterministic turing machine

$O(n^k)$ $k \geq 1$

NP = The set of problems that can be solved using a non-deterministic turing machine in polynomial time & once the solution is found it can be verified in polynomial time

NP-Hard = A problem P is NP-hard if a problem K in the set of NP-complete can be reduced to P in polynomial time

NP-Complete = the hardest NP-hard problems in NP

NP-hard:
P1: find the smallest element in an array
P2: sorting
P1 is reduced to P2 since it need help from another problem

TSP problem:

TSP Decision: given a graph G & a weight W, does G have a path that is at most W?

TSP Search: is a decision problem where you return the path(s) w/ at most W.

TSP Optimization: find the minimum path TSP

(1) is the problem decidable ✓
(2) is the problem solvable in polynomial time ($O(n^k)$) ✗
(3) once a solution is found, can you verify the solution in polynomial time $O(n^k)$ ✓

Therefore TSP Decision = NP-complete. Same w/ TSP Search
TSP Optimization is not (1), (2) or (3)

$T(n) = 9T\left(\frac{n}{2}\right) + n + 1; \quad T(0) = 1;$

The theta complexity of this complexity function is undefined

Answer: True ✓

```
function f(n)
If n > 1 Then
x = 2 * f(n/2)
print("final exam")
x = 2 * f(n/2)
```

Answer O(n) ✓

What is the big O time complexity (worst case) of the following pseudocode if we assume that $n \geq 1$?

```
FUNCTION f(n):
   INITIALIZE i = 0, k = 0, p = 0
   FOR index i to (n-1) DO
      FOR index j = 2 to index i DO
         PRINT("Final Exam")
         j = (j * j)
      ENDFOR
      i = i + 1
   ENDFOR
   WHILE (k < n) DO
      WHILE (p > 0) DO
         p = p + 1
      ENDWHILE
      k = k + 1
      PRINT("Final Exam")
   ENDWHILE
   PRINT("Final Exam")
ENDFUNCTION
```

- ○ $O(n)$
- ✓ ● $O(n \log \log n)$
- ○ None of the answers given.
- ○ $O(n^2)$

The following adjacency matrix is given for the Traveling Salesman Problem where the values in the matrix represent the distances between adjacent vertices.

	V1	V2	V3	V4	V5
V1	0	34	32	23	15
V2	21	0	30	25	31
V3	18	33	0	44	42
V4	10	24	67	0	21
V5	90	30	33	31	0

Using a Dynamic Programming approach, What is the bound of the path $g(V3, \{V2, V4\})$ if we assume that the beginning and ending of the shortest path is vertex $V1$?.

- ○ 60
- ○ 89
- ○ None of them
- ✓ ● 68

$$4T\left(\frac{n}{2}\right) + \frac{n^2}{\log n} \in \Theta\left(n^2 \log_2(\log_2 n)\right)$$

- ✓ ● True
- ○ False

Given the following definitions covered in class for the topic of NP=P:

- P = all the decision problems that can (always) be solved in polynomial time and which solutions can also be verified in polynomial time
- NP = all the decision problems that may (Yes or No) be solved in polynomial time and which solutions can also be verified in polynomial time

The Divide and Conquer algorithm implemented in homework #2 is in the set of P problems.

- ✓ ● True
- ○ False

Asymptotic order

- In order to be more specific about limiting bounds, the following notations are also used: $o(g(n))$ *and* $\omega(g(n))$ representing small o-notation and small omega-notation
- Comparison
  - $f(n) \in O(g(n)) \rightarrow f(n) \leq g(n)$
  - $f(n) \in o(g(n)) \rightarrow f(n) < g(n)$
  - $f(n) \in \Omega(g(n)) \rightarrow f(n) \geq g(n)$
  - $f(n) \in \omega(g(n)) \rightarrow f(n) > g(n)$
  - $f(n) \in \Theta(g(n)) \rightarrow f(n) = g(n)$

How to provide proof that a function is bounded to a given specific asymptotic notation? For example, prove that $n^2 + 3n + 2 \in O(n^3)$

Limit Ratio Notation Table

$O(g(n))$	$f(n) \leq g(n)$	$< \infty$
$o(g(n))$	$f(n) < g(n)$	zero
$\Omega(g(n))$	$f(n) \geq g(n)$	> 0
$\omega(g(n))$	$f(n) > g(n)$	∞
$\Theta(g(n))$	$f(n) = g(n)$	$\mathbb{R} > 0$ and $< \infty$

If no Log $P=0$

$$2^x = 8 \rightarrow 2^3 = 8$$

$t(n) = 8 + \left(\frac{n}{2}\right) + n^3$

Master theorem

$a = 8$

$b = 2$ $\quad \log_6 a = \log_2 8 = 3$

$k = 3$ $\quad \Theta(n^3 \log n)$

$P = 0$

$t(n) = 2T\left(\frac{n}{2}\right) + \frac{n}{\log n} \rightarrow n\log^{-1} n$

$a = 2$

$b = 2$ $\quad \log_6 a = \log_2 2 = 1 = k$

$K = 1$

$P = -1$ $\quad \Theta\left(n \log(\log n)\right)$

$t(n) = 8t\left(\frac{n}{4}\right) + n^2 + 1$

$4^2 = 16$

$a = 8$

$b = 4$ $\quad \log_4 8 < k$ $\quad 1 \leq \log_4 8 \leq 2$

$k = 2$ $\quad \Theta(n^2 \log^0 n)$

$P = 0$ $\quad \Theta(n^2)$

$t(n)$ $3T\left(\frac{n}{4}\right) + n$

$a = 3$ $\quad \log_4 3 < k$

$b = 4$ $\quad \Theta\left(n^k \log^P n\right)$

$k = 1$ $\quad \Theta(n)$

$P = 0$