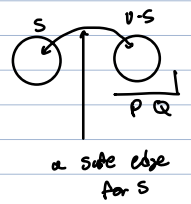
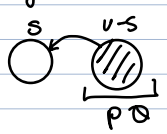
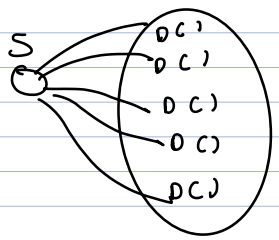
Prim's Alg



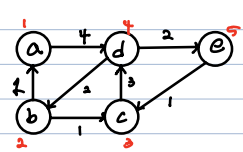
Dijkstra Alg





11/28(목)

Floyd's Alg – All pairs shortest distances (paths)



D

	a	b	c	d	e
a	0	∞	∞	4	∞
b	1	0	1	∞	∞
c	∞	∞	0	3	∞
d	∞	2	∞	0	2
e	∞	∞	1	∞	0

D^0

```
for k = 1 to n do
  for i = 1 to n do
    for j = 1 to n do
      D[i,j] = min(D[i,j], D[i,k] + D[k,j])
```
(D[i,j]: previous stage)

$D^{(1)\to a}$

	a	b	c	d	e
a	0	∞	∞	4	∞
b	1	0	1	5	∞
c	∞	∞	0	3	∞
d	∞	2	∞	0	2
e	∞	∞	1	∞	0

$D^{(2)\to b}$ (1, 2 고정, 나중 변함)

	a	b	c	d	e
a	0	∞	∞	4	∞
b	1	0	1	5	∞
c	∞	∞	0	3	∞
d	3	2	3	0	2
e	∞	∞	1	∞	0

$D^{(3)\to c}$

	a	b	c	d	e
a	0	∞	∞	4	∞
b	1	0	1	4	∞
c	∞	∞	0	3	∞
d	3	2	3	0	2
e	∞	∞	1	4	0

$D^{(4)\to d}$

	a	b	c	d	e
a	0	6	∞	4	6
b	1	0	1	4	6
c	6	5	0	3	5
d	3	2	3	0	2
e	7	6	1	4	0

$D^{(5)\to e}$

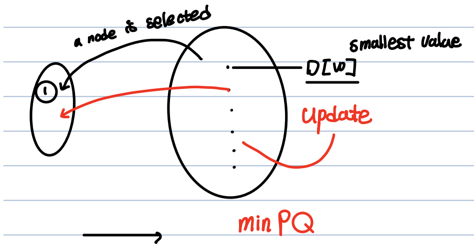
	a	b	c	d	e
a	0	6	∞	4	6
b	1	0	1	4	6
c	6	5	0	3	5
d	3	2	3	0	2
e	7	6	1	4	0

$D^{k-1}(i,j)$ $D^{k-1}(i,k) + D^{k-1}(k,j)$

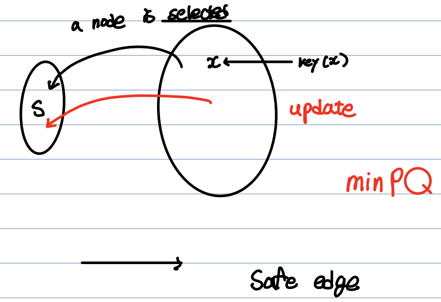
$D^k(i,j)$ $\min(D, D)$

D^3 D^4 D^5

Dijkstra's Alg (G, w) ← positive real #S



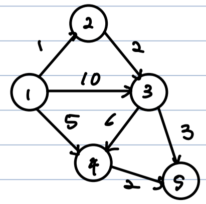
Prim's Alg (G, w, s)



D(2) :
D(3)
⋮
D(n)

1
2
3

3 ① sel ② update
9



D(2) = 1
D(3) = 10
D(4) = 5
D(5) = ∞

---

11 / 29 / 화

p.673 다익스트라 → Greedy 전략

Dijkstra's Alg (G, w) n nodes 가중방향 w 그래프 R^+

1 S = {1} → initialization
2 δ = ∅ D[ ] ‖ c[i,j] = edge cost > 0
3 Q = G.V

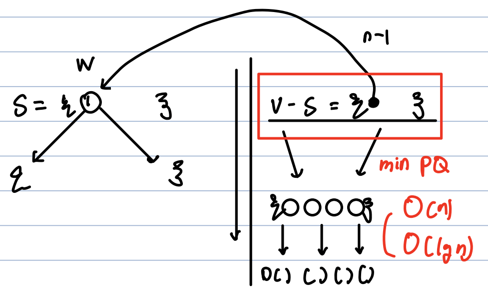
4 for i = 1 to n-1 do
5 ① selection w {__}
6 ② update min
7 D( ) min(D( ), 0+0)
8
9

D[2] = ○
D[3] = ○
⋮
D[n] = ○

|V|번 반복
updated

Similarity with other Algorithm
① Prim's Alg (Selection) → min, plus
② Floyd's Alg (update) → {∞} ∪ {0} ∪ R^+
Warshall's Alg → {0, 1} (or, and)

D(i,j) = min (D( ), 0 ⊕ 0)
$R^k(i,j)$ = or ($R^{k-1}(i,j)$, $R^{k-1}(i,k)$ and ...)

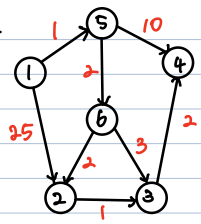


a semiring structure

1 2)

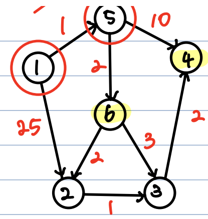
제일 작은 w 선택

Ex



D(2) = 25
D(3) = ∞ → 1 로부터 direct node가 없다.
D(4) = ∞
D(5) = 1
D(6) = ∞

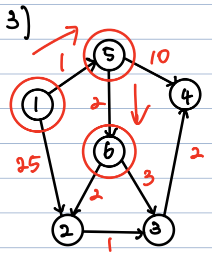
S = {1} V-S = {2, 3, 4, 5, 6}



S = {1, 5}
D(5) = 1

V-S : {2, 3, 4, 6}
D(2) = 25
D(3) = ∞
D(4) : 11
D(6) = 3

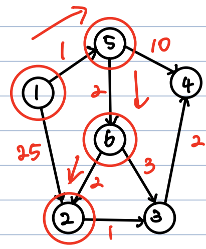
3)



제일 작은 w 선택

S = {1, 5, 6}
D(5) = 1
D(6) = 3

V-S : {2, 3, 4}
D(2) = 5
D(3) = 6
D(4) : 11



제일 작은 w 선택

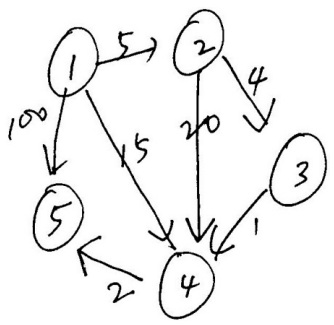
S = {1, 5, 6, 2}
D(5) = 1
D(6) = 3
D(2) : 5

V-S : {3, 4}
D(3) = 6
D(4) : 8

O(V) lg V
O(V) V번
O(V lg V)
↓
각 Extract min 연산
이진 최소힙

	출발점	노드(간선) 간 가중치	Data structure
Prim's Alg			
Floyd			
Dijkstra	O (Greedy)	O (모든 양수)	PQ (Prime Queue : 우선순위 큐)

```
1  S = {1}
2  for i = 2 to n do
3     D[i] = C[1,i]      // initialization
4  for i= 1 to n-1
5     choose a vertex w in V - S such that
6        D[w] is a minimum
7     add w to S
8     for each vertex v in V - S
9        D[v] = min(D[v], D[w]+C[w,v])
```



$D(2) = c(1,2) = 5$

$D(3) = c(1,3) = \infty$

$D(4) = c(1,4) = 15$

$D(5) = c(1,5) = 100$

initialization.

$S = \{1\}$ $V - S = \{2, 3, 4, 5\}$

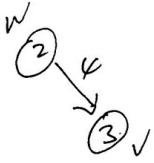
$D(2)$ is min, therefore,

$S = \{1, 2\}$, $V - S = \{3, 4, 5\}$

update.

$D(v) = \min(D(v), D(w) + c[w,v])$

$D(w) = 2$



$c(w,v) = 4$

~~D(3)~~ $D(3) = 9$

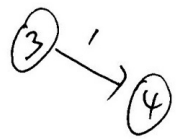
$D(4) = 15$

$D(5) = 100$

```
1  S = {1}
2  for i = 2 to n do
3     D[i] = C[1,i]      // initialization
4  for i= 1 to n-1
5     choose a vertex w in V - S such that
6          D[w] is a minimum
7     add w to S
8     for each vertex v in V - S
9          D[v] = min(D[v], D[w]+C[w,v])
```

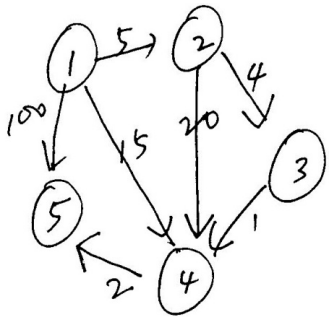
$S = \{1,2,3\}$, $V - S = \{4,5\}$

$D(v) = \min(D(v), D(w) + c(w,v))$



$c(3,4) = 1$ $D(4) = 10$

$D(5) = 100$



shortest dist.

2	5
3	9
4	10
5	12

$S = \{1,2,3,4\}$, $V - S = \{5\}$

$D(5) = \min(D(5), D(4) + c(4,5))$

$D(5) = 12$

Skip

Transitive closure,
all-pairs shortest distances,
single source shortest distances

Set, Operations

$$R^k(i,j) = \vee\left(R^{k-1}(i,j), \wedge\left(R^{k-1}(i,k), R^{k-1}(k,j)\right)\right)$$

{0,1}, AND, OR

$$D(i,j) = \min\left(D(i,j), +\left(D(i,k), D(k,j)\right)\right)$$

$$p(v) = \min\left(D(v), +\left(D(w), c(w,v)\right)\right)$$

Non-negative real numbers,
infinity, MIN, PLUS

skip

def $a^* = a^0 + a^1 + a^2 + \cdots$

A closed semiring

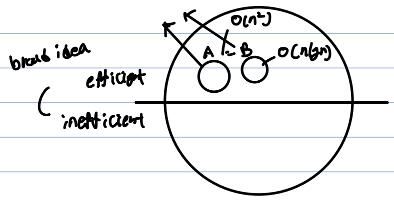
Definition 3.1 *An algebraic structure* $C = (S, +, \cdot, ^*, 0, 1)$ *is called* ***closed semiring*** *if and only if it fulfills the following conditions:*

1. S *is a set;* $+$ *and* $\cdot$ *are two binary operators called* ***addition*** *and* ***multiplication****, respectively;* * *is a unary operator called* ***closure****; and* 0 *and* 1 *are elements of* S
2. *For every* $a, b, c \in S, a + (b + c) = (a + b) + c$*; addition is associative.*
3. *For every* $a, b \in S, a + b = b + a$*; addition is commutative.*
4. *For every* $a \in S, a + 0 = a$*; that is,* 0 *is a neutral element for addition.*
5. *For every* $a, b, c \in S, a \cdot (b \cdot c) = (a \cdot b) \cdot c$*; multiplication is associative.*
6. *For every* $a \in S, a \cdot 1 = 1 \cdot a = a$*; that is,* 1 *is a neutral element for multiplication.*

   $a \cdot 0 = 0 \cdot a = 0$
7. *For every* $a, b, c \in S, a \cdot (b + c) = a \cdot b + a \cdot c$*, and* $(a + b) \cdot c = a \cdot c + b \cdot c$*; multiplication distributes over addition.*
8. *For every* $a \in S, a^* = 1 + a \cdot a^* = 1 + a^* \cdot a$. □

# of steps – time
worst time

Ins sort ( )
O(n)
$O(n^2)$



# of steps (worst case)

$O(n^k)$
polynomial time Alg

P25

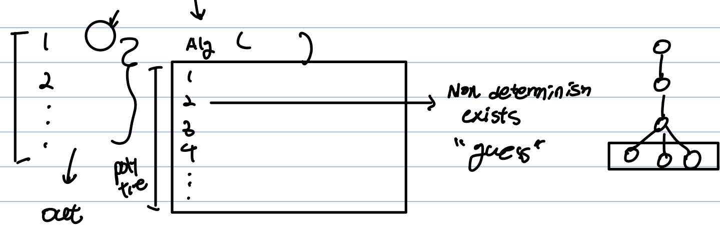
ch 34

1) P = { x | ◯ ∃ an efficient alg for x } — 다항시간안에 풀수있는 문제
↑ a decision problem | Y 1 . . / N 0
"Easy" problems

2) NP = { x | ______ } ① ② 2 properties for NP – 다항시간에 "확인" 할수 있는 문제들

① NP = { x | ∃ an efficient non-deterministic alg } → prob → –DNA computer, – Q computer) Von Neumann comp

a (deterministic) alg



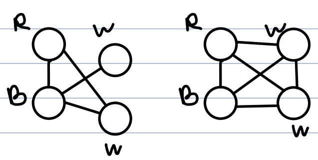
현실적으로 쓸모없는 (ex $O(n^{100})$) 알고리즘이 발견되면, 종종 더 효율적인 알고리즘이 등장한다.

② NP = { x | ∃ a "clue" w, if the Answer for this problem is Yes then verification is done efficiently }
= witness
= Certificate

Ex 3 coloring problem ∈ NP

input: an undirected graph $G = (V, E)$

Output: if G is 3-colorable then Y, else N

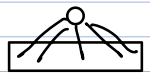


Y N

(1) property

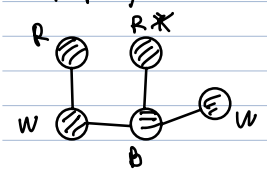
Alg (G) n nodes

1. Guess a coloring C. ⟶ 3^n possibility



2. if C satisfies the constraint then return Yes else N

Property "clue"



1. O(V)
2. for each pair of nodes $O(V^2)$

Efficient algorithms?

certain classes of problems. An algorithm is deemed efficient if it can solve a problem in polynomial time, which means the running time of the algorithm is a polynomial function of the length of the input. There are classes of harder problems for which the fastest possible algorithm requires exponential time. Another criterion is the space requirement of the algorithm. There is a crucial distinction between algorithms that can find a solution, verify a solution, or list several distinct solutions in given time and space. The complexity hierarchy that is generated in this way is the foundation of theoretical computer science. Precise complexity results can be notoriously difficult. The famous question whether polynomial time equals nondeterministic polynomial time (i.e., P = NP) is one of the hardest open problems in computer science and all of mathematics. Here, we consider simple

NP, reduction, NP hard

The class nondeterministic polynomial time (denoted as NP) consists of problems for which solutions exist that are of polynomial length, and given a candidate for a solution of polynomial length, whether the candidate is indeed a solution can be checked in polynomial time. Therefore, an NP algorithm can verify a solution in polynomial time.

To proceed further, we need the notion of "reduction" between classes of problems. A reduction, from a given problem P_1 to a problem P_2, is a translation such that a solution for P_2 can provide a solution for P_1. More precisely, if there is a polynomial-time reduction from P_1 to P_2, then a polynomial-time algorithm for P_2 implies a polynomial-time algorithm for P_1.

A given problem is NP-hard if for every problem in NP, there is a polynomial reduction to the given problem. A problem is NP-complete if it is both NP-hard and there is an NP algorithm for the problem.

Ch 26 Reduction

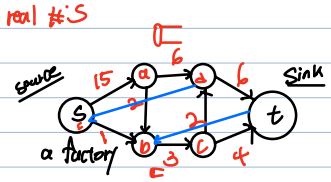
① max flow problem — a directed graph

② max bipartite graph matching problem — an undirected graph

ch 26.1

Def a flow network
- a directed graph with some property

pds. real life



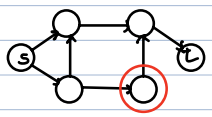
products "capacities"

send products
s to t
공장에서 도착점

a directed graph
① for each pair of nodes x, y $c(x,y) \geq 0$
capacities
$c(a,d) = 6$
$c(d,b) = 0$: 간선이 없으면 0

② 2 special nodes, s, t
(s): No incoming edges.
(t): No outgoing edges.

③ each node except s, t should be on any path s → t



X Not possible

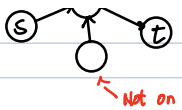
12/6/화

pg.7

a flow network
- a directed graph
① source : where the products are constantly provided. | source (s) ⇝ (t) sink
② Sink : destination (where products go) |

- Each nodes (except s, t) should be on a path from s to t.

Ex)

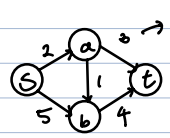


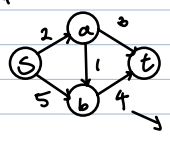
(X)

$c(u,v) \in R^+$, ~~$c(u,u)$~~

- For each edge (u,v), a capacity

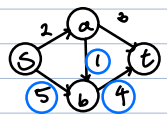
Ex of flow network and Dijkstra's Alg





(G, s)

Dijkstra's



def: a flow f in G $\quad f: V \times V \to R$

① capacity constraint: $f(x,y) \le c(x,y)$

② skew symmetry: $f(x,y) = -f(y,x)$

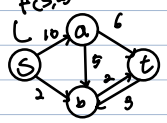
③ flow conservation:

Except s,t

capacity

capacity constraint $f(s,a) \le 10$

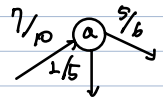
$f(a,b) \le 5$



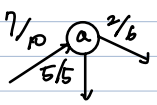
$f(s,b) = 1.5$

$f(b,s) = -1.5$

} skew symmetry



flow value $\le$ capacity



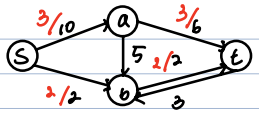
1st problem:

Maximum flow Problem

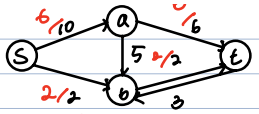
input: a flow net

Output: max of $|f|$

Ex1 Ex2



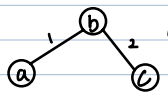
$|f| = 5$



$|f| = 8$

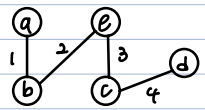
an undirected graph $G=(V,E)$

def: a matching in G is a subset of E such that $\forall_{x \in V}$ at most 1 edge in M touches x.
(x is incident on at most 1 edge in M)



a, b - adjacent
edge 1 is incident on a, b

Ex



- $\emptyset$: 0
- $\{1,2\}$ X (touches 3 edges)
- $\{1,4\}$ a b c d e ○
- $\{1,2,3\}$ X
- $\{2\}$: 0

Max matching.

$\{1,3\}$
$\{2,4\}$

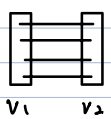
---

an undirected graph $G=(V,E)$

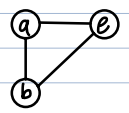
def: a bipartite graph
(2 / division)

$V = V_1 \cup V_2$ $V_1 \neq \emptyset$ $V_2 \neq \emptyset$

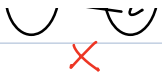
all Edges in E



Ex

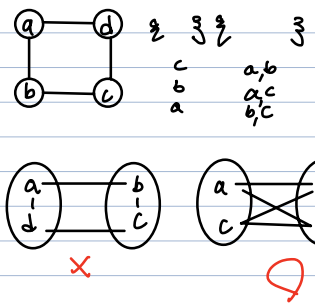


$\{\}$ $\{c\}$ $\{b\}$ $\{a\}$ $\{a,b\}$ $\{a,c\}$ $\{b,c\}$



×

Ex2



2nd problem:

Max bipartite graph Matching

input: a bipartite graph
Output: |§ ∃| max

1st problem:

Maximum flow Problem A

input: a flow net
Output: max of |f|

A B

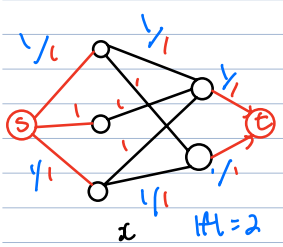
reduction

~~A ≤ B~~ A reduces to B

B ≤ A B reduces to A

B, A

As long as A is solvable, B is solvable



∀x

1. An arbitrary instance of B

2. x is converted an instance of A

Max matching

{1, 3}
{2, 4}

A ***flow network*** $G = (V, E)$ is a directed graph in which each edge $(u, v) \in E$ has a nonnegative ***capacity*** $c(u, v) \geq 0$. If $(u, v) \notin E$, we assume that $c(u, v) = 0$. We distinguish two vertices in a flow network: a ***source*** s and a ***sink*** t. For convenience, we assume that every vertex lies on some path from the source to the sink. That is, for every vertex $v \in V$, there is a path $s \rightsquigarrow v \rightsquigarrow t$. The graph is therefore connected, and $|E| \geq |V| - 1$. Figure 26.1 shows an example of a flow network.

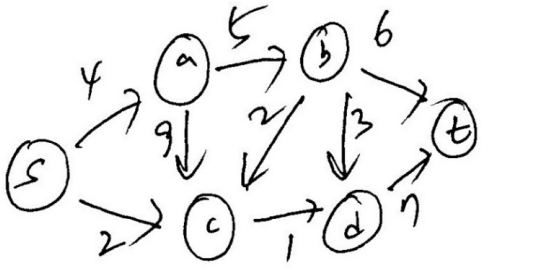
We are now ready to define flows more formally. Let $G = (V, E)$ be a flow network with a capacity function c. Let s be the source of the network, and let t be the sink. A ***flow*** in G is a real-valued function $f : V \times V \to \mathbf{R}$ that satisfies the following three properties:

Capacity constraint: For all $u, v \in V$, we require $f(u, v) \leq c(u, v)$.

Skew symmetry: For all $u, v \in V$, we require $f(u, v) = -f(v, u)$.

Flow conservation: For all $u \in V - \{s, t\}$, we require

$$\sum_{v \in V} f(u, v) = 0 .$$

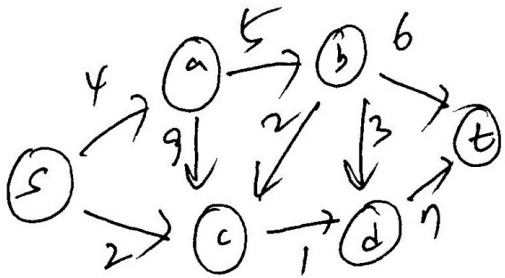


a flow network

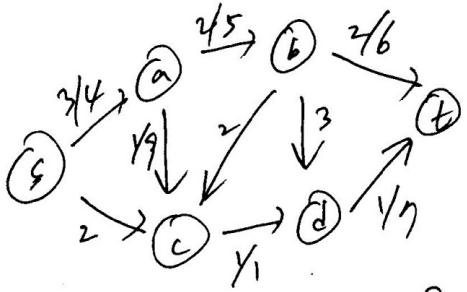
capacity constraint – the value of a flow f cannot exceed a capacity of an edge
for example, it is possible to send a flow whose value is 3 from a to b, but it is impossible to send a flow whose value is 3 from c to d

skew symmetry – if the flow value from a to c is 7, then the flow value from c to a is -7

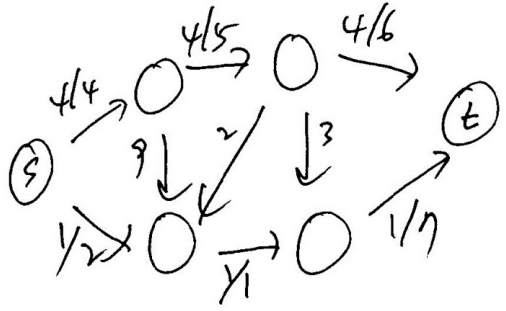
flow conservation – if the amount of flow value for the flow to a is 3, then the amount of the flow value that comes out of a is 3 as well



a flow network



3



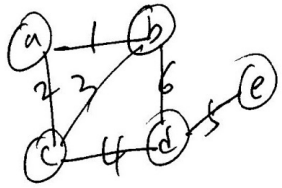
5

3

5

Maximum flow problem – given a flow network, find a flow whose value is maximum

Given an undirected graph $G = (V, E)$, a ***matching*** is a subset of edges $M \subseteq E$ such that for all vertices $v \in V$, at most one edge of M is incident on v. We say that a vertex $v \in V$ is ***matched*** by matching M if some edge in M is incident on v; otherwise, v is ***unmatched***. A ***maximum matching*** is a matching of maximum cardinality, that is, a matching M such that for any matching M', we have $|M| \geq |M'|$.



G

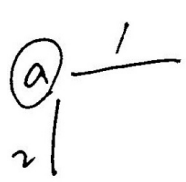
a matching

{1}

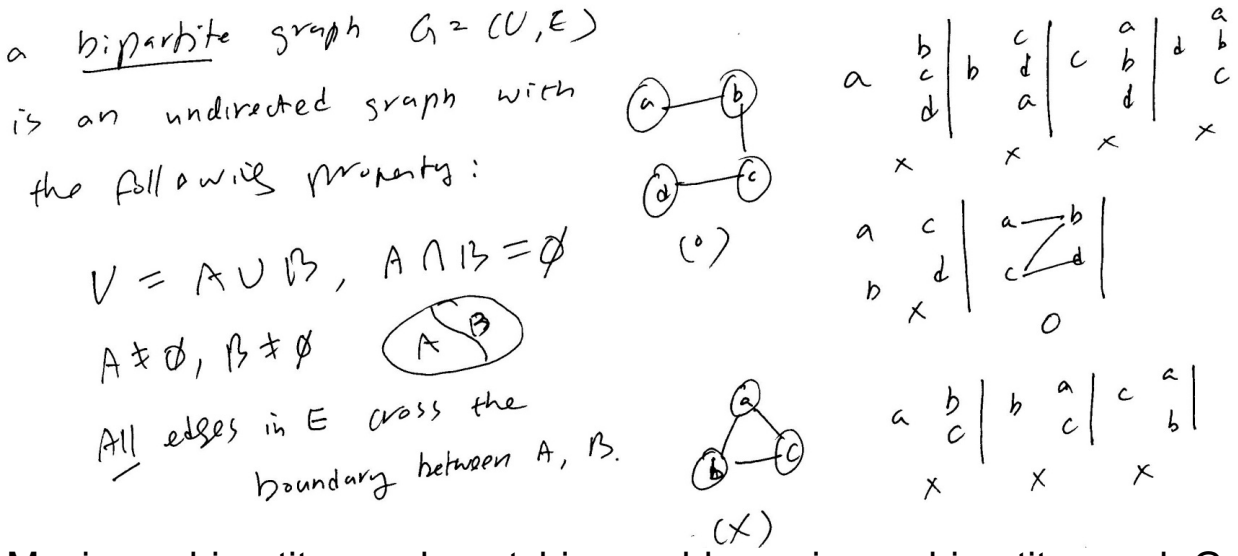
{1, 4}

{2, 6}

...



edges 1, 2 are incident on a "touching"



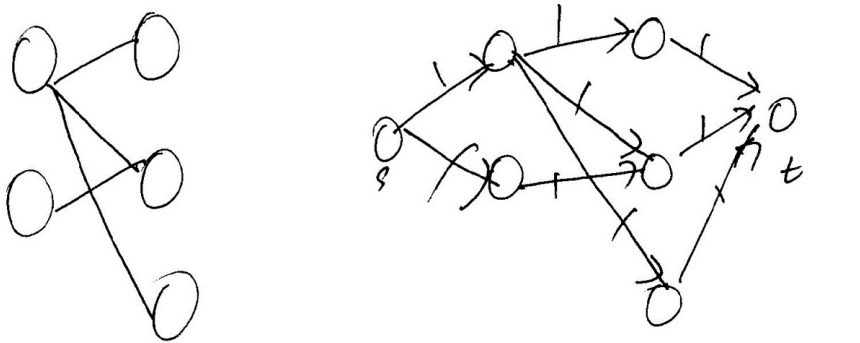
Maximum bipartite graph matching problem: given a bipartite graph G, what is the maximum number of a matching in G (or, find such a matching)

Maximum bipartite graph matching problem is reducible to maximum flow problem

What does this mean?

It is always possible to convert an arbitrary instance x of the Maximum bipartite graph matching problem to an instance y of the maximum flow problem in such a way that an answer for y can be used to find an answer to x (note: the direction is important)

Their structures are similar!



max flow value
= 2

max #
of a matching
= 2

Ch29. Linear Programming Problem (LPP)

optimization

a tabular method

LPP ∈ P

Polynomial time Algs
- ~~Interior Point Alg~~
- ~~Ellipsoid Alg~~

In practice ↓ Not used, Simplex Alg is used instead.

"most" instances → performance of "Simplex Alg" is very good.

$O(2^n)$ X

QS Alg : $O(n \lg n) — O(n^2)$

Linear Programming Problem

input: [① a linear function f in n variables; ② finite # of Linear Constraints]

Ex

$f(x_1, x_2, x_3) = 4x_1' - 2x_2' + 3x_3'$ — linear

a linear constraint — $\{f() \le 0\}$ / $\{\_\_\_ \ge 0\}$

$x_1 + 2x_3 + 4x_2 \le 0$

$2x_1 + 4 \ge 0$

Output: max f (min f)

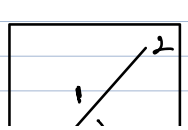
Example

$f(x_1, x_2) = 2x_1 + 3x_2$

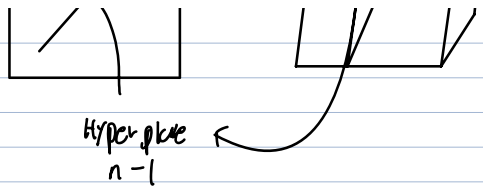
max f

subject to $x_1 - x_2 \ge 0$, $x_1 + x_2 \le 1$

2D plane



2D


Hyperplane
n-1

Chapter 29. linear programming

a linear program consists of a linear function that needs to be optimized (= maximized, or minimized) subject to a finite number of linear inequalities [it is assumed that variables take values from real numbers]

Let us first consider the following linear program with two variables

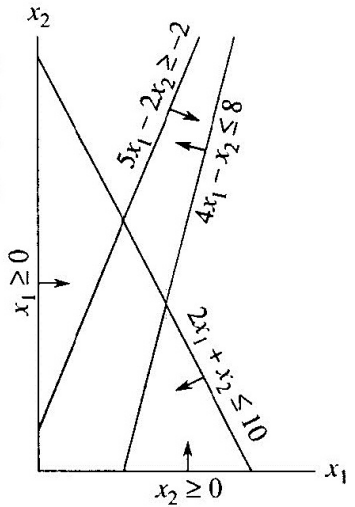
$$
\begin{array}{lrcrcr}
\text{maximize} & x_1 & + & x_2 & & \\
\text{subject to} & & & & & \\
& 4x_1 & - & x_2 & \leq & 8 \\
& 2x_1 & + & x_2 & \leq & 10 \\
& 5x_1 & - & 2x_2 & \geq & -2 \\
& x_1, x_2 & & & \geq & 0 \ .
\end{array}
$$

2 variables → possible to draw graph

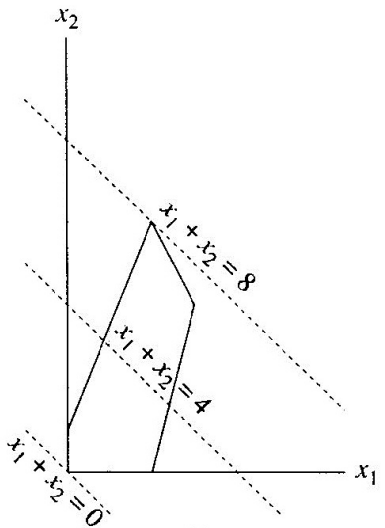
maximize $x_1 + x_2$

subject to

$$\begin{aligned} 4x_1 - x_2 &\le 8 \\ 2x_1 + x_2 &\le 10 \\ 5x_1 - 2x_2 &\ge -2 \\ x_1, x_2 &\ge 0 \end{aligned}$$



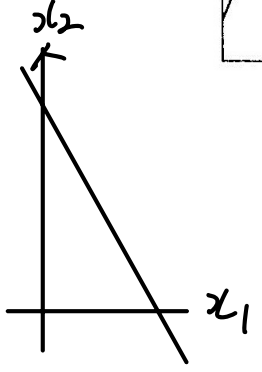
(a)



(b)

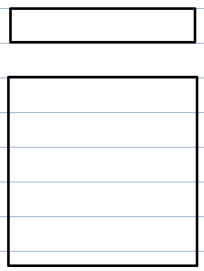
$x_1 + x_2 = k$

$x_2 = k - x_1$



P16 Standard form

a line pnz



objective form

12/7/wed

LP (linear program) in Standard form

$$\text{maximize} \quad \sum_{j=1}^{n} c_j x_j$$

objective function

$$\text{subject to}$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad \text{for } i = 1, 2, \ldots, m$$

$$x_j \geq 0 \quad \text{for } j = 1, 2, \ldots, n$$

m+n constraints

minimize $-2x_1 + 3x_2$

subject to

$$\begin{aligned} x_1 + x_2 &= 7 \\ x_1 - 2x_2 &\le 4 \\ x_1 &\ge 0 , \end{aligned}$$

change to

$x_1 + x_2 \le ?$

$x_1 + x_2 \ge ?$

and we negate the coefficients of the objective function, we obtain

maximize $2x_1 - 3x_2$

subject to

$$\begin{aligned} x_1 + x_2 &= 7 \\ x_1 - 2x_2 &\le 4 \\ x_1 &\ge 0 . \end{aligned}$$

Simplex Algorithm

conversion LP into slack form

Real numbers

maximize $2x_1 - 3x_2 + 3x_3$

subject to

$$\begin{aligned} x_1 + x_2 - x_3 &\le 7 \\ -x_1 - x_2 + x_3 &\le -7 \\ x_1 - 2x_2 + 2x_3 &\le 4 \\ x_1, x_2, x_3 &\ge 0 \end{aligned}$$

standard form

x_1, x_2, x_3 linear constraint

x_4, x_5, x_6 : slack variables

the name of the objective function

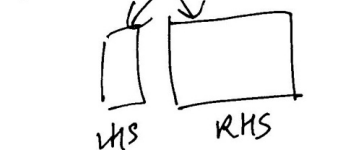
slack form

$$\begin{aligned} z &= 2x_1 - 3x_2 + 3x_3 \\ x_4 &= 7 - x_1 - x_2 + x_3 \\ x_5 &= -7 + x_1 + x_2 - x_3 \\ x_6 &= 4 - x_1 + 2x_2 - 2x_3 \end{aligned}$$

$x_1, x_2, x_3, x_4, x_5, x_6 \ge 0$

non-basic variables

basic variables



slack form

$0 \le x_4 \Leftrightarrow x_1 + x_2 - x_3 \le 7$

$0 \le x_5 \Leftrightarrow -x_1 + x_2 + x_3 \le -7$

$0 \le x_6 \Leftrightarrow x_1 - 2x_2 + 2x_3 \le 4$

How to convert LP into a new LP

29.3 simplex algorithm

standard form

$$\begin{aligned}
\text{maximize} \quad & 3x_1 + x_2 + 2x_3 \\
\text{subject to} \quad & \\
& x_1 + x_2 + 3x_3 \le 30 \\
& 2x_1 + 2x_2 + 5x_3 \le 24 \\
& 4x_1 + x_2 + 2x_3 \le 36 \\
& x_1, x_2, x_3 \ge 0
\end{aligned}$$

Non basic variables

slack form

$$\begin{aligned}
z &= 3x_1 + x_2 + 2x_3 \\
x_4 &= 30 - x_1 - x_2 - 3x_3 \\
x_5 &= 24 - 2x_1 - 2x_2 - 5x_3 \\
x_6 &= 36 - 4x_1 - x_2 - 2x_3
\end{aligned}$$

Basic variables

select a non-basic variable and change its position with one basic variable [LHS – RHS]

largest coefficient. x_1

$x_4 \ge 0 \rightarrow x_1 \le 30$

$x_5 \ge 0 \rightarrow x_1 \le 12$

$x_6 \ge 0 \rightarrow x_1 \le 9$

x_6

x_1 is selected first bc C_1 is the biggest.

↳ $0 \le x_4 \rightarrow x_1$ cannot exceed 30

Smallest, so we choose x_6

plug in slack form

before

$$
\begin{aligned}
z &= 3x_1 + x_2 + 2x_3 \\
x_4 &= 30 - x_1 - x_2 - 3x_3 \\
x_5 &= 24 - 2x_1 - 2x_2 - 5x_3 \\
x_6 &= 36 - 4x_1 - x_2 - 2x_3
\end{aligned}
$$

after

$$
\begin{aligned}
z &= 27 + \frac{x_2}{4} + \frac{x_3}{2} - \frac{3x_6}{4} \\
x_1 &= 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4} \\
x_4 &= 21 - \frac{3x_2}{4} - \frac{5x_3}{2} + \frac{x_6}{4} \\
x_5 &= 6 - \frac{3x_2}{2} - 4x_3 + \frac{x_6}{2}
\end{aligned}
$$

$x_6 = 36 - 4x_1 - x_2 - 2x_3$

$4x_1 = 36 - x_2 - 2x_3 - x_6$

$x_1 = 9 - \frac{1}{4}x_2 - \frac{1}{2}x_3 - \frac{1}{4}x_6$

$z = 3(9 - \frac{1}{4}x_2 - \frac{1}{2}x_3) - \frac{1}{4}x_6 + x_2 + 2x_3$

$= 27 - \frac{3}{4}x_2 - \frac{3}{2}x_3 - \frac{1}{4}x_6 + x_2 + 2x_3$

$= 27 + \frac{1}{4}x_2 + \frac{1}{2}x_3 - \frac{1}{4}x_6$

$x_4 = 30 - (9 - \frac{1}{4}x_2 - \frac{1}{2}x_3 - \frac{1}{4}x_6) - x_2 - 3x_3$

$= 21 - \frac{3}{4}x_2 - \frac{5}{2}x_3 + \frac{x_6}{4}$

$x_5 = 24 - 2(9 - \frac{1}{4}x_2 - \frac{1}{2}x_3 - \frac{1}{4}x_6) - 2x_2 - 5x_3$

$= 6 - \frac{3x_2}{2} - 4x_3 + \frac{x_6}{2}$

x_2 係數 $\frac{1}{4}, -\frac{3}{4}$

$x_3 \leq \frac{21}{5}$

$x_3 \leq \frac{6}{4}$

choose x_5

again, select a non-basic variable → x_3
and exchange its position with one basic variable

before

$$
\begin{aligned}
z &= 27 + \frac{x_2}{4} + \frac{x_3}{2} - \frac{3x_6}{4} \\
x_1 &= 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4} \\
x_4 &= 21 - \frac{3x_2}{4} - \frac{5x_3}{2} + \frac{x_6}{4} \\
x_5 &= 6 - \frac{3x_2}{2} - 4x_3 + \frac{x_6}{2}
\end{aligned}
$$

basic variables
non-basic variables

$x_3 = 6 - \frac{3}{2}x_2 + \frac{1}{2}x_6 - x_5$

$x_3 = \frac{3}{2} - \frac{3}{8}x_2 + \frac{1}{8}x_6 - \frac{1}{4}x_5$

after

$$
\begin{aligned}
z &= \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \\
x_1 &= \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \\
x_3 &= \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \\
x_4 &= \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16}
\end{aligned}
$$

$$
\begin{aligned}
z &= 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2x_6}{3} \\
x_1 &= 8 + \frac{x_3}{6} + \frac{x_5}{6} - \frac{x_6}{3} \\
x_2 &= 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3} \\
x_4 &= 18 - \frac{x_3}{2} + \frac{x_5}{2} .
\end{aligned}
$$

at this stage, there does not exist a non-basic variable
whose coefficient is positive, max value = 28 and each variable has a value
8, 4, 0, 18, 0, 0, respectively.

pivot operation : convert an LP into a different LP
such that the optimal value remains the same
it is iteratively executed until ALL non-basic variables have NEGATIVE coefficients