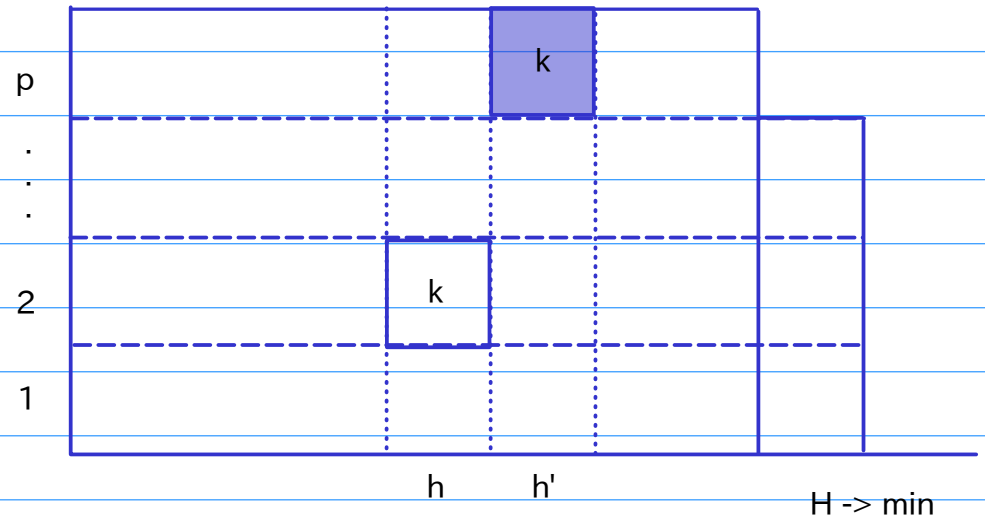
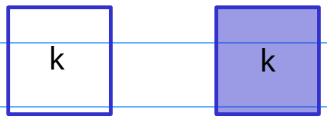
On the mini report #6

Q: Suppose we have a frying pan that can cook at most p steaks at the same time and it takes T time to cook one side of a steak. What is the optimal (i.e., minimizing the ending time) way to cook n steaks?

Observation:



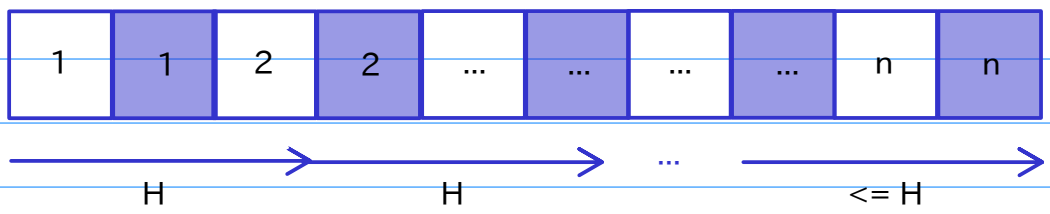


front and back
of the k^th steak

constraint:

$h(k) \neq h'(k)$ for all $k = 1, 2, \ldots, n$

How to find a minimum H such that no white and black blocks of the same number stay in the same level? A simple idea is to arrange white and black blocks of the same number side by side serially.

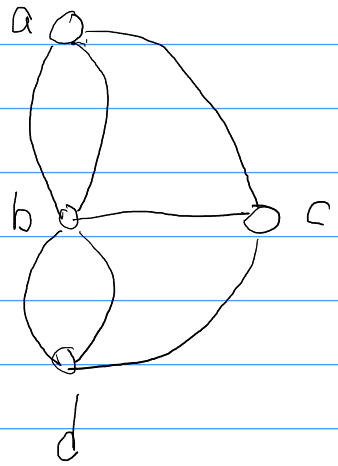


Let $H = \max\{2, \lceil \frac{2n}{p} \rceil\}$

It can be seen that layout in HT time is possible and it is optimal (notice that HT is a lower bound on the makespan).

graph data representation (review)

$$n = |V|,\ m = |E|$$



incidence matrix

	a	b	c	d
a	0	2	1	0
b	2	0	1	2
c	1	1	0	1
d	0	2	1	0

Pro: easy

Con: inefficient for sparse graph

adjacent lists

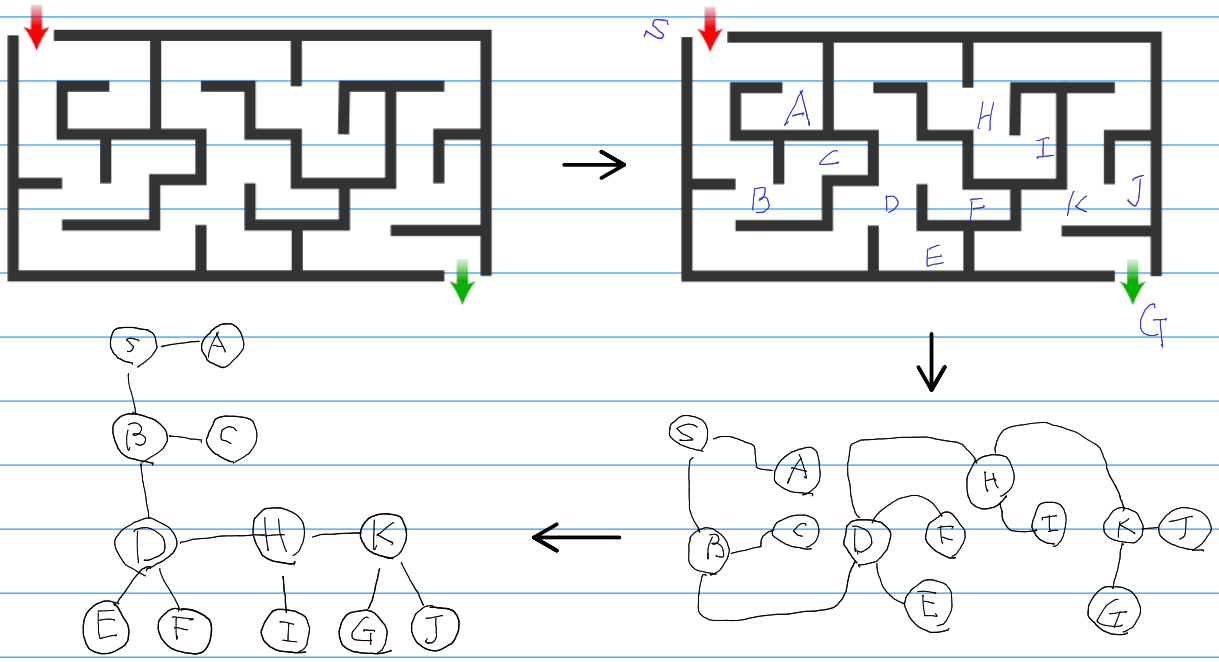
a	b	b	c		
b	a	a	c	d	d
c	a	b	d		
d	b	b	c		

Pro: compact

Con: complicate

Depth first search (DFS)

https://en.wikipedia.org/wiki/Hedge_maze



solve the maze = find a path from the start to the goal

The next algorithm was stated by Charles Pierre Tremaux (1858-1882), called Tremaux's algorithm in robot motion.

DFS (Depth-First Search)

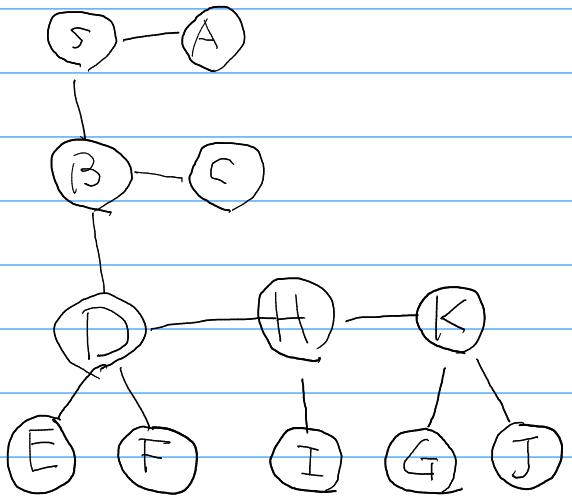
INPUT: G=(V,E) given by adjacent lists, nodes s and t
OUTPUT: an s,t-path (or its nonexistence)

```
main
    parent[s] = s
    parent[v] = -1 for all v <> s
    DFS(s)


DFS(v)
    for all neighbors w of v {
        if parent[w] == -1 {
            parent[w] = v
            DFS(w)
        }
    }
```

time: O(m+n)
space: O(m+n)

Demo (exercise)



adjacent lists

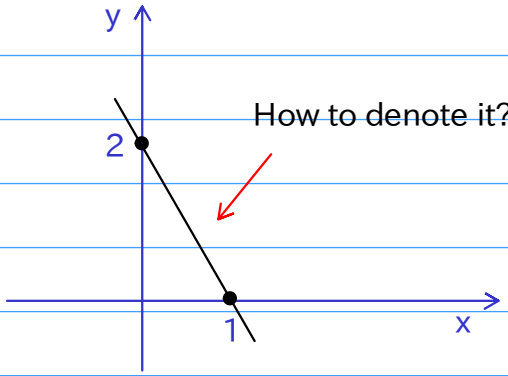
A - S
B - C - D - S
C - B
D - B - E - F - H
E - D
F - D
G - K
H - D - I - K
I - H
J - K
K - G - H - J
S - A - B

	A	B	C	D	E	F	G	H	I	J	K	S
Parent												
DFS(v)												

Q. How to output the s-t path?

Remark: graph can be arbitrary; t = none => search all nodes

Straight line, hyperplane, and half space



In general (in n-dimensional space)

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

is an n-1 dimension space called a hyperplane.

Half spaces separated by the hyperplane.

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \geq b$$

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b$$

Mathematical Programming

$$\min\{f(x)|x \in \Omega \subseteq \mathbb{R}^n\}$$

* f: object function

* x: (decision) variable

* constraint(s): $x \in \Omega \subseteq \mathbb{R}^n$

Note: min{ f } = max{ -f }, thus min or max is only technical.

Ex. Linear Programming (LP), the standard formulation

$$c_1x_1 + c_2x_2 + \cdots + c_nx_n \rightarrow \max$$

subject to (s.t.) $a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2$$

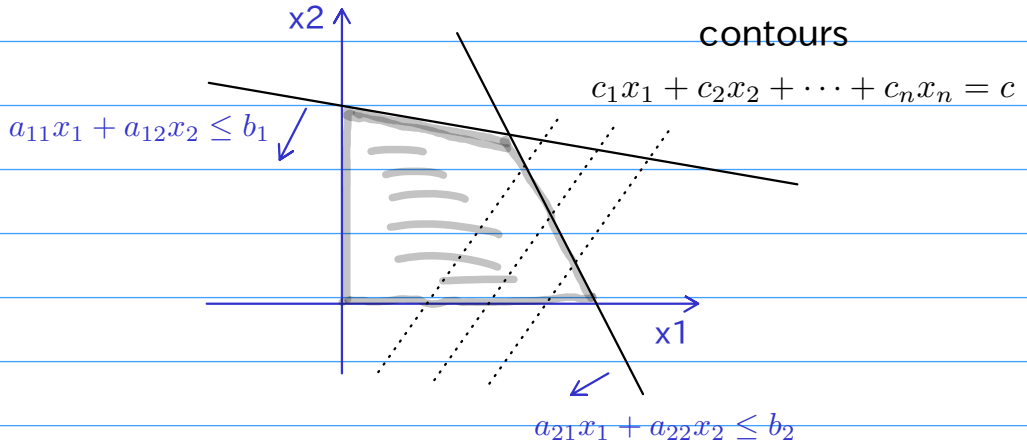
$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m$$

$$x_i \geq 0 \quad i = 1, 2, \ldots, n$$

#48

LP on plane (n=2)



Notations

* vector: $x = (x_i) \in \mathbb{R}^n = \mathbb{R}^{n\times 1}$

* matrix: $A = (a_{ij}) \in \mathbb{R}^{m\times n}$

Let

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n, \quad c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \in \mathbb{R}^n, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \in \mathbb{R}^m$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \in \mathbb{R}^{m\times n}$$

$$\Rightarrow \text{LP} \quad \max \{ c^T x \mid Ax \le b, \ x \ge 0 \}$$

Solve LP (or IP, Integer Programming) problems

Simplex, Interior point, etc., but usually we use a solver.

e.g.,

https://online-optimizer.appspot.com/ -> default model

On Mini Report: Taro's problem

$$x_i = \begin{cases} 1 & \text{buy menu } i \\ 0 & \text{don't buy menu } i \end{cases} \qquad 1 \le i \le n$$

$$\text{price } c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \qquad \text{requirement } b = \begin{pmatrix} -2.0 \\ -1.0 \\ \vdots \\ 2.5 \end{pmatrix}$$

$$\text{coefficient } A = \begin{pmatrix} -R^T \\ -G^T \\ \vdots \\ S^T \end{pmatrix}$$

Use "var x1 binary;" to specify x1 is either 0 or 1, etc.