#37

On the mini report #5

* Game theory in practice

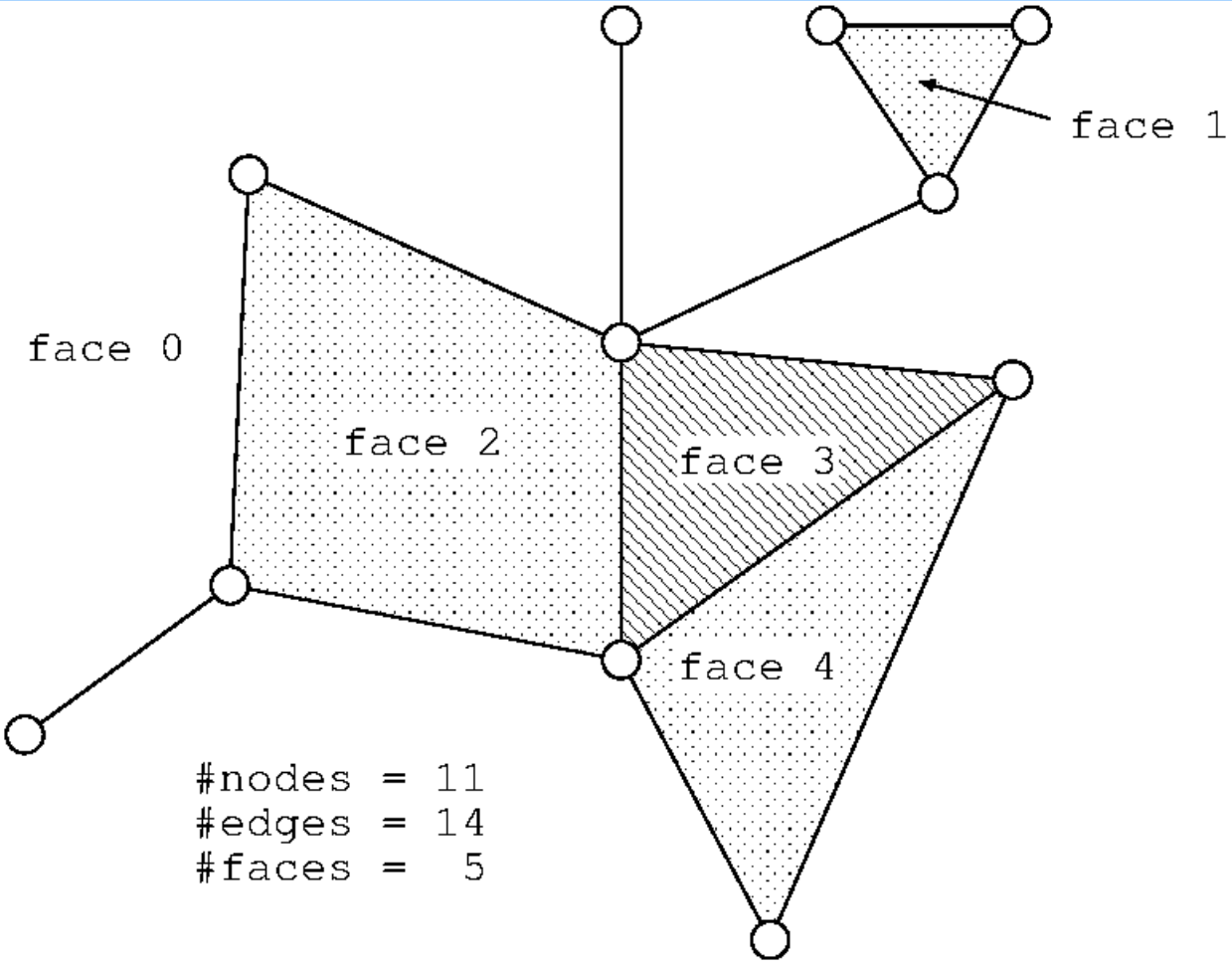
* Euler Theorem (n + f - m = 2)

Proof

Step 1. It holds for a tree since f = 1 and m = n - 1 as proved in lecture #5.

Step 2. Suppose that it holds for all connected plane graph with k face(s).

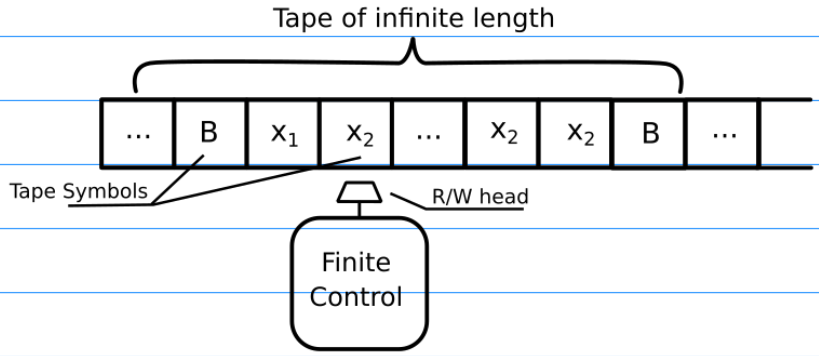
Step 3. Consider a connected plane graph with k+1 face(s). Let F be a face of it. Remove one edge from F. The new graph G' is still a connected plane graph but with k face(s) (why?). Thus, by the assumption of Step 2, n' + f' - m' = 2. Since n'=n, f' = f - 1, and m' = m - 1, we have n +f -m = n' + f' - m' = 2. //



#38

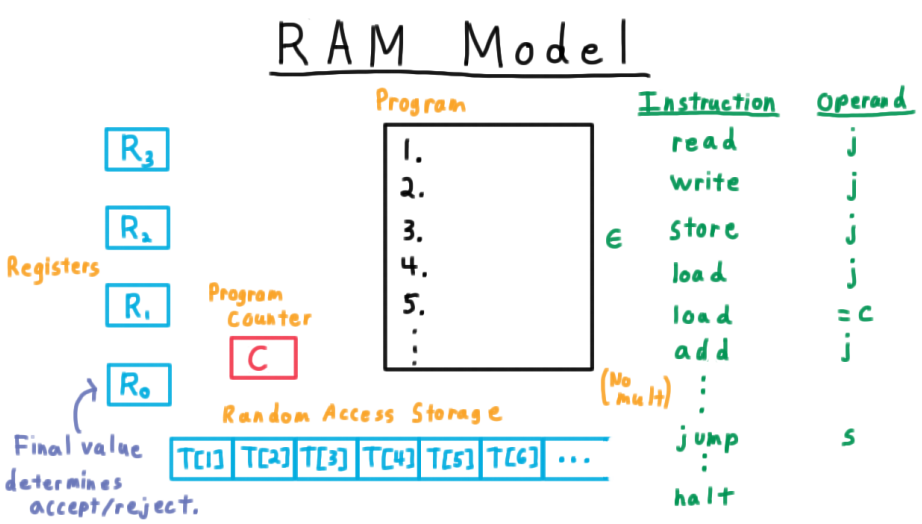
What is "computation"? (Computation models)

1. Turing Machine



Picture credit: https://iq.opengenus.org/general-introduction-to-turing-machine/

2. Random Access Machine (RAM)



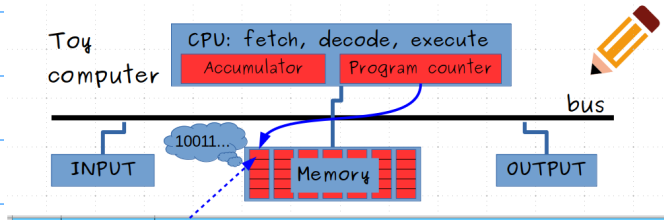
https://s3.amazonaws.com/content.udacity-data.com/courses/gt-cs6505/churchturing.html

Two factors in designing algorithms

1. How the data are stored
2. How to process those data

N. Wirth 1976 (Turing Award 1984)

Data Structures + Algorithms = Programs



label	instruction	description
	get	get a number from keyboard into accumulator
L	print	print contents of accumulator
	load Val	load accumulator with Val (Val unchanged)
	store M	store contents of accumulator into memory location called M
	add Val	add Val to contents of accumulator (Val unchanged)
	sub Val	subtract Val from contents of accumulator (Val unchanged)
	goto L	go to instruction labeled L
	ifpos L	go to instruction labeled L if accumulator is >= zero
	ifzero L	go to instruction labeled L if accumulator is zero
	stop	stop running
M	Num	before program runs, set this memory location (called M) to Num

Illustration of a toy computer designed by B.W. Kernighan

Computational ability (polynomial time or not): Turing machine = RAM

Our daily computer differs from the RAM model mainly by the limited memory.

Inside a computer (10') => https://www.youtube.com/watch?v=ExxFxD4OSZ0

Three fundamental structures of an algorithm

Sequence	step 1; step 2; ...
Conditional	IF ... ELSE ... END
Loop	WHILE ... DO ... END

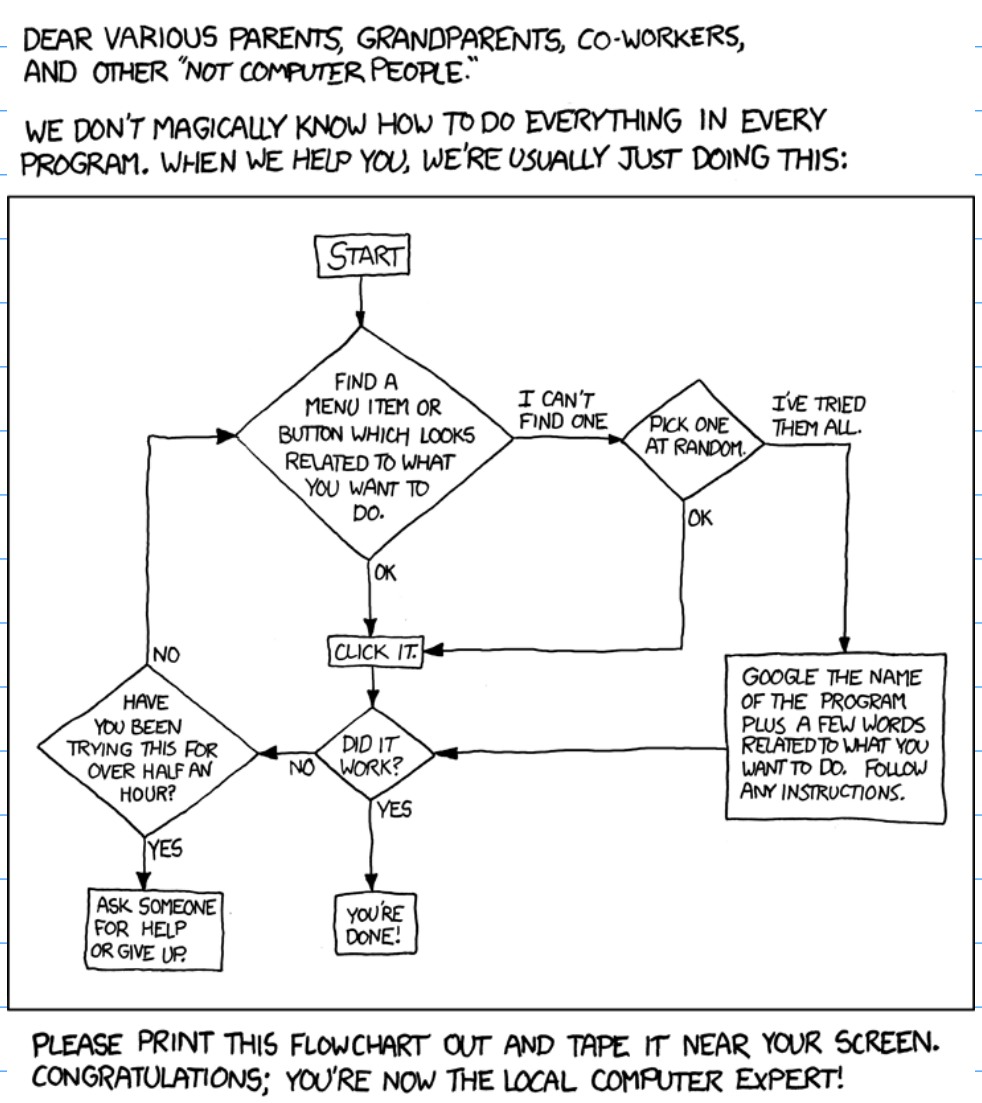


Figure credit: https://xkcd.com/627/

Data structure illustrated

Cooking hamburger steaks

* We want to cook some steaks using one frying pan.
* Cooking each side of a steak takes two minutes.
* Each time at most two steaks can be cooked.

=> How long does it take to cook n steaks?

picture: https://www.tasteofhome.com/article/pan-fried-burgers/

Simple stupid algorithm (SSA):

```
for all steaks s {
        cook (both sides of) s and output
}
```

=> time complexity: 4n

space complexity: 0

Simple algorithm (SA): an improvement

```
for i = 1, 2, ..., ⌊n/2⌋ {
        cook two steaks at the same time and output them
}
if n > 2 * ⌊n/2⌋ { // i.e., n is an odd number
        cook the n^th steak and output it
}
```

$\lfloor n/2 \rfloor$: max integer <= n/2

=> time complexity: 2n for an even n and 2n+2 for an odd n

space complexity: 0

Clever algorithm (CA): (use an additional plate)

if i = 1 or n = 2 * $\lfloor n/2 \rfloor$ {

Apply SA to cook all steaks

}

else {

Apply SA to cook n-3 steaks

Exercise

}

=> time complexity: 4 for n=1; 2n otherwise (optimal. Why?)

space complexity: 1 for odd n>=3; 0 otherwise

Machine scheduling

Cooking example

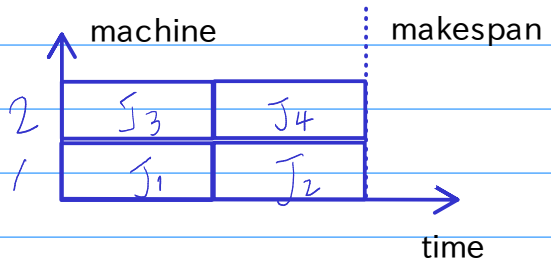
Each time at most two steaks can be cooked => two machines M1, M2

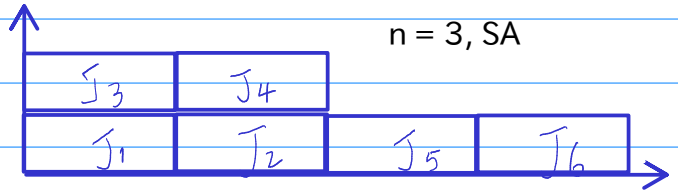
n steaks with two sides => Jobs: (two sides of steak 1) J1, J2, (steak 2) J3, J4, ...

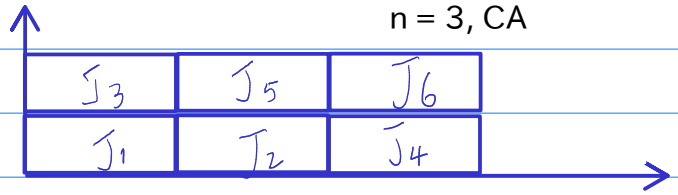
Cooking each side takes two minutes. => Processing time = 2 minutes for all jobs.

Constraint: Jobs J_{2i-1} and J_{2i} cannot be processed at the same time for all i.

e.g., n = 2

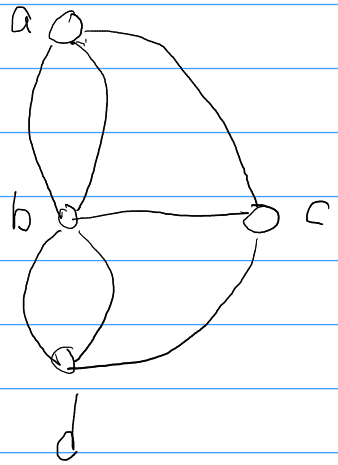






Data representation for graphs

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



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

size n^2

adjacent lists

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

Pro: compact

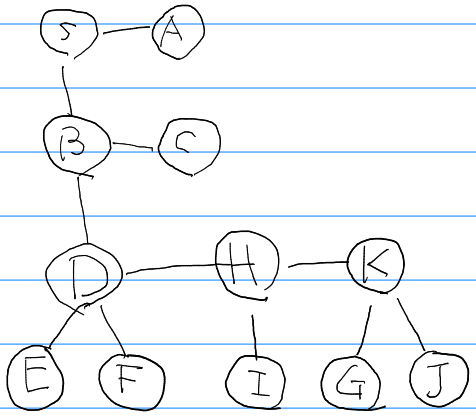
Con: complicate

size

undirected $n + 2m$

directed $n + m$

Exercise



incidence matrix

adjacent lists