COMPUTES A STABLE MATCHING

GALE - SHAPLEY ()

- INITIALLY, EACH $a_i \in A$ AND EACH $b_j \in B$ IS FREE
- WHILE THERE EXISTS A FREE $a_i \in A$ THAT HAS NOT YET PROPOSED TO ALL $b_j \in B$:
  - LET $a_i$ BE A FREE ITEM OF A.
  - LET $B' \subseteq B$ BE THE SET OF THE $b_j$'S THAT $a_i$ HAS NOT YET PROPOSED TO.
  - LET $b_j \in B'$ BE THE ELEMENT OF $B'$ THAT RANKS HIGHEST IN $a_i$'S PREFER. LIST
  - IF $b_j$ IS FREE:
    - MATCH UP $a_i$ AND $b_j$ (WHO ARE THEN NOT FREE ANYMORE)
  - ELSE:
    - LET $a_k$ BE THE CURRENT PARTNER OF $b_j$
    - IF $b_j$ PREFERS $a_k$ TO $a_i$:
      - NOTHING CHANGES ($a_i$ REMAINS FREE, AND $\{b_j, a_k\}$ REMAIN TOGETHER)
    - ELSE:
      - "BREAK UP" $\{b_j, a_k\}$
      - "MATCH" $\{b_j, a_i\}$
      - $a_k$ BECOMES FREE.

$\leq n^2$ ITERATIONS

ANALYZING AN OPTIMIZATION PROBLEM

- YOU WANT TO UNDERSTAND IT PRECISELY
- YOU WANT TO GUESS AN ALGORITHM FOR SOLVING IT
- " " " PROVE THAT IT SOLVES THE PROBLEM.
- TRY TO REDUCE THE RUNTIME OF YOUR ALGORITHM.

---

"EFFICIENCY" ?

DEF ??: " AN ALGORITHM IS EFFICIENT IF IT RUNS QUICKLY ON INSTANCES".

"QUICKLY" CAN MEAN:
"FEW SECONDS"
"ONE DAY"
" $O(n^2)$ "
" $O(n)$ "

---

WORST-CASE ANALYSIS

WE WANT THE RUNTIME OF THE ALGORITHM TO BE BOUNDED BY THE SIZE OF THE (WORST) INPUT.

THEN, IN GALE-SHAPLEY'S CASE, THE NUMBER OF ITERATIONS IS AT MOST $n^2$.

$a_1, \dots, a_n$ $\quad$ $b_1, \dots, b_n$

SUPPOSE, NOW, THAT WE DOUBLE THE "SIZE" OF A GALE - SHAPLEY INSTANCE:

$a_1, \dots, a_N$ $\quad$ $b_1, \dots, b_N$ $\quad$ WITH $N = 2n$.

THEN THE NUMBER OF ITERATIONS BECOMES

$$N^2 = (2n)^2 = 4 \cdot n^2$$

$\left( N = c \cdot n \implies N^2 = (cn)^2 = c^2 n^2 \right.$. THUS, IF $c$ IS A CONSTANT, THE RUNTIME BLOWUP IS ONLY CONSTANT)

NOW, LET'S CONSIDER THE MOST TRIVIAL ALGORITHM (TEST EACH OF THE $n! = n \cdot (n-1) \cdots 2 \cdot 1$ PERFECT MATCHINGS).

THIS ALGORITHM CONSIDERS $n!$ PERF. MATCHING WHEN THERE ARE $n$ A'S AND $n$ B'S.

IF $N = 2n$, THE RUNTIME BECOMES

$$N! = (2n)! = 2n\,(2n-1)(2n-2)\cdots(n+1)\cdot n\,(n-1)\cdots 3 \cdot 2 \cdot 1$$
$$= 2n\,(2n-1)\cdots(n+1)\cdot n!$$
$$\geq n^n\, n!$$

---

DEF (?): AN ALGORITHM HAVING A RUNTIME $\leq c \cdot n^d$, WHERE $c$ AND $d$ ARE CONSTANTS, ON INPUTS OF SIZE $n$ IS SAID TO BE A POLYNOMIAL-TIME ALGORITHM (POLYTIME ALGORITHM).

IF I HAVE AN ALGORITHM RUNNING IN TIME $\leq c \cdot n^d$ ON INPUTS OF SIZE $n$, IF I RUN THAT ALGORITHM ON AN INPUT OF SIZE $N = b \cdot n$, THEN

$$c \cdot N^d = c\,(b \cdot n)^d = c \cdot b^d \cdot n^d$$

$n \log \log n$ VS $\boxed{n^7}$

$$\log \log n \leq 7$$
$$\Updownarrow$$
$$\log n \leq 10^7$$
$$n \leq 10^{10^7}$$

| | $n$ | $n \log_2 n$ | $n^2$ | $n^3$ | $1.5^n$ | $2^n$ | $n!$ |
|---|---|---|---|---|---|---|---|
| $n = 10$ | < 1 sec | < 1 sec | < 1 sec | < 1 sec | < 1 sec | < 1 sec | 4 sec |
| $n = 30$ | < 1 sec | < 1 sec | < 1 sec | < 1 sec | < 1 sec | 18 min | $10^{25}$ years |
| $n = 50$ | < 1 sec | < 1 sec | < 1 sec | < 1 sec | 11 min | 36 years | very long |
| $n = 100$ | < 1 sec | < 1 sec | < 1 sec | 1 sec | 12,892 years | $10^{17}$ years | very long |
| $n = 1{,}000$ | < 1 sec | < 1 sec | 1 sec | 18 min | very long | very long | very long |
| $n = 10{,}000$ | < 1 sec | < 1 sec | 2 min | 12 days | very long | very long | very long |
| $n = 100{,}000$ | < 1 sec | 2 sec | 3 hours | 32 years | very long | very long | very long |
| $n = 1{,}000{,}000$ | 1 sec | 20 sec | 12 days | 31,710 years | very long | very long | very long |

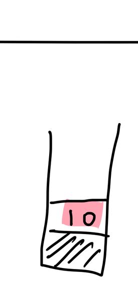
---

DEF.: AN ALGORITHM HAS A WORST-CASE RUNTIME OF $f(n)$ IF THE ALGORITHM NEVER MAKES MORE THAN $f(n)$ OPERATIONS ON INPUTS OF SIZE $n$.

---

```
DEF INC(A):
    RETURN A + 1

INC(10)
...
```



① STACK

② RUN INC, LOADING "A" INTO THE NAME SPACE

③ IT PUSHES 10 OUT OF THE STACK AND INTO A'S MEMORY LOCATION

④ LOAD A'S VALUE INTO A CPU REGISTER, SUM IT UP TO 1, AND LOAD THE RESULT ONTO THE STACK

```
DEF INC(A):
    RETURN A + 1

B = 0
FOR i IN RANGE(n):
    B = INC(B)
```

LET $\alpha$ BE THE NUMBER OF OPERATIONS REQUIRED BY INC (A)

LET $\beta$ BE THE NUMBER OF OPERATIONS REQUIRED BY "B = 0"

LET $\gamma$ ~~BE~~ THE NUMBER OF OPERATIONS REQUIRED BY A RUN OF THE LOOP

OUR ALGORITHM, THEN, TAKES TIME

$$\beta + n(\gamma + \alpha) = c \cdot n + c'$$

$c = \gamma + \alpha$
$c' = \beta$

---

$O$, $\Omega$, $\Theta$ NOTATIONS

TO AVOID GETTING LOST INTO CONSTANTS, WE DISREGARD THEM.

SUPPOSE THAT $T(n)$ IS THE (WORST-CASE) RUNTIME OF AN ALGORITHM $(T(n) = 135n^2 + 15n + 3)$

DEF. IF $T(n)$, AND $f(n)$, ARE NON-NEGATIVE INCREASING FUNCTIONS, WE SAY THAT "$T(n)$ IS $O(f(n))$" IF $\exists\, c, n_0 > 0$ SUCH THAT $T(n) \leq c \cdot f(n)$ $\forall n \geq n_0$.

$T(n) = O(f(n))$
$T(n) \leq O(f(n))$