O, Ω, Θ NOTATION

DEF: IF $T(n)$ AND $f(n)$ ARE TWO NON-NEGATIVE, INCREASING, FUNCTIONS, WE SAY THAT "$T(n)$ IS A $O(f(n))$" ($T(n) \le O(f(n))$) ($T(n) = O(f(n))$) IF $\exists\ c, n_0 > 0$ S.T.

$T(n) \le c \cdot f(n) \quad \forall n \ge n_0$.

$T(n) = p\, n^2 + q\, n + r$ WITH $p, q, r \ge 0$ AND $p > 0$.

$f(n) = n^2$

THEN $\forall n \ge n_0 = 1$ IT HOLDS THAT

$q \cdot n \le q \cdot n^2$ AND

$r \le r \cdot n^2$.

THUS, $T(n) = p \cdot n^2 + q \cdot n + r \le p \cdot n^2 + q \cdot n^2 + r \cdot n^2$

$= (p + q + r) \cdot n^2 = c \cdot n^2$ (WITH $c = p + q + r$)

AND $T(n) \le O(f(n)) = O(n^2)$.

---

① "$100\ n^2 \le O(n^3)$" ? YES

$T(n) = 100 \cdot n^2 \le 100 n^3 \quad \forall n \ge 1 = n_0$

$f(n) = n^3$ WITH $c = 100$ AND $n_0 = 1$

WE HAVE THAT $100 n^2 \le c \cdot f(n)\ \forall n \ge n_0$

② "$\frac{1}{100} n^2 \le O(n^{1.9})$" ? × NO

③ "$n = O(2^n)$" ? YES

④ "$n = O(1.1^n)$" ? YES

$\left(\lim_{n \to \infty} \frac{1.1^n}{n} = \infty\right)$

$T(n) = 100 n \le O(n^2) \le O(n^3)$

$T(n) = O(n)$

DEF: IF $T(n)$ AND $f(n)$ ARE TWO NON-NEGATIVE, INCREASING, FUNCTIONS, WE SAY THAT "$T(n)$ IS A $\Omega(f(n))$" ($T(n) \ge \Omega(f(n))$) ($T(n) = \Omega(f(n))$) IF $\exists\ c, n_0 > 0$ S.T.

$T(n) \ge c \cdot f(n) \quad \forall n \ge n_0$.

$T(n) = 100 \cdot n$

$T(n) \le O(n^2)$

$T(n) \le O(n)$

$T(n) \ge \Omega(n)$

$T(n) \not\ge \Omega(n^2)$

DEF: IF $T(n) \le O(f(n))$ AND $T(n) \ge \Omega(f(n))$, THEN $T(n) = \Theta(f(n))$.

$T(n) = n^2 = O(n^2)$

$T(n) = \Omega(n)$ ← TRUE BUT SUBOPTIMAL

$T(n) = \Omega(n^2)$ ← TRUE AND OPTIMAL

$T(n) = \Omega(n^3)$ ← FALSE

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 $a_i \in A$: ← $\le n^2$ ITERATIONS
  - 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.

① IDENTIFY A FREE a_i

② IDENTIFY THE b_j THAT
  (I) RANKS HIGHEST IN a_i'S PREFERENCE LIST, AND THAT
  (II) a_i HASN'T YET PROPOSED TO.

③ DETERMINE WHETHER b_j IS CURRENTLY FREE AND, IF NOT, HER CURRENT PARTNER a_k

④ DETERMINE WHETHER b_j PREFERS a_i TO a_k

IF WE CAN IMPLEMENT ①, ②, ③ AND ④ IN $O(1)$ (CONSTANT) TIME, THEN AN ITERATION TAKES $O(1)$ TIME. GIVEN THAT THE NUMBER OF ITERATIONS IS $\le n^2$, THE TOTAL RUNTIME WOULD THEN BE $n^2 \cdot O(1) = O(n^2)$.

TO OBTAIN THESE $O(1)$ IMPLEMENTATIONS, WE MAKE USE OF SPECIAL DATA STRUCTURES.

Ⓐ APREF: A $n \times n$ ARRAY (MATRICE) SUCH THAT APREF[i][l] CONTAINS THE INDEX OF THE l^{TH} MOST PREFERRED PARTNER OF a_i.

$a_3 : b_2 > b_1 > b_4 > b_3$

APREF[3][1] = 2
APREF[3][2] = 1
APREF[3][3] = 4
APREF[3][4] = 3

Ⓑ NEXT: AN ARRAY OF SIZE n, SUCH THAT NEXT[i] CONTAINS THE RANK (IN a_i'S PREFERENCE LIST) OF THE NEXT b_j THAT a_i IS GOING TO PROPOSE TO.

AT THE OUTSET, WE SET NEXT[i] = 1 $\forall i$.

WHEN a_i NEEDS TO PROPOSE, HE'LL PROPOSE TO b_j WHERE j = APREF[i][NEXT[i]].

AFTER A PROPOSAL OF a_i WE SET NEXT[i] = NEXT[i] + 1.

THEN, APREF AND NEXT LET US IMPLEMENT ② IN $O(1)$ TIME.

Ⓒ CURRENT, AN ARRAY OF SIZE n, SUCH THAT CURRENT[j] CONTAINS THE INTEGER i SUCH THAT b_j IS CURRENTLY ENGAGED TO a_i. IF b_j IS CURRENTLY FREE, CURRENT[j] = NONE.

THEN, CURRENT ALLOWS US TO IMPLEMENT ③ IN $O(1)$ TIME.

Ⓓ RANKING, AN $n \times n$ ARRAY, SUCH THAT RANKING[j][i] IS THE RANK OF a_i IN b_j'S PREFERENCE LIST.

$b_3 : a_2 > a_1 > a_3 > a_4$

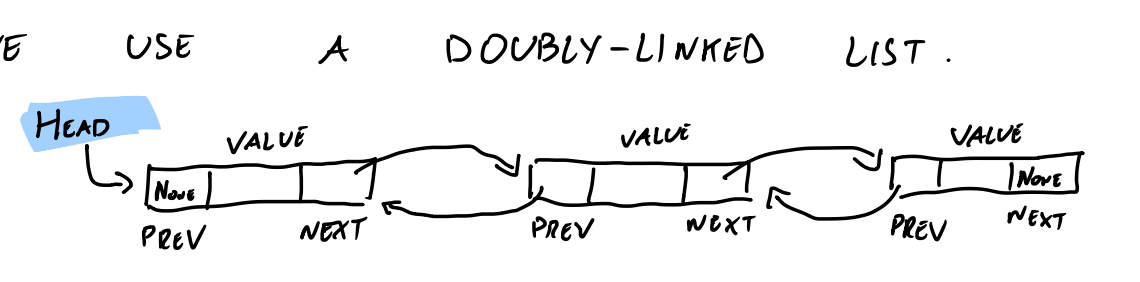
RANKING[3][2] = 1
RANKING[3][1] = 2
RANKING[3][3] = 3
RANKING[3][4] = 4

THEN, (BY CHECKING WHETHER RANKING[j][i] < RANKING[j][k]) WE CAN TELL WHETHER b_j LIKES a_i MORE THAN a_k IN $O(1)$ TIME. SO ④ TAKES $O(1)$ TIME.

THEN, ②, ③ AND ④ TAKE $O(1)$ TIME.

WHAT ABOUT ①?

WE USE A DOUBLY-LINKED LIST.



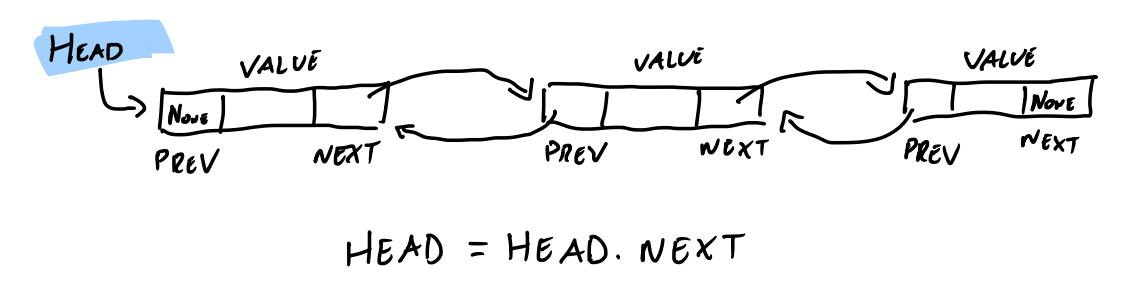
TO GET THE FIRST VALUE OF THE LIST:

HEAD.VALUE

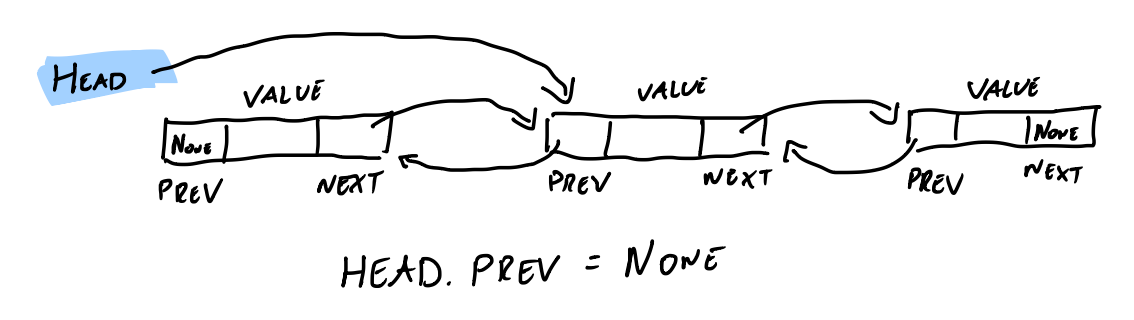
TO GET THE VALUE OF THE SECOND NODE:

HEAD.NEXT.VALUE

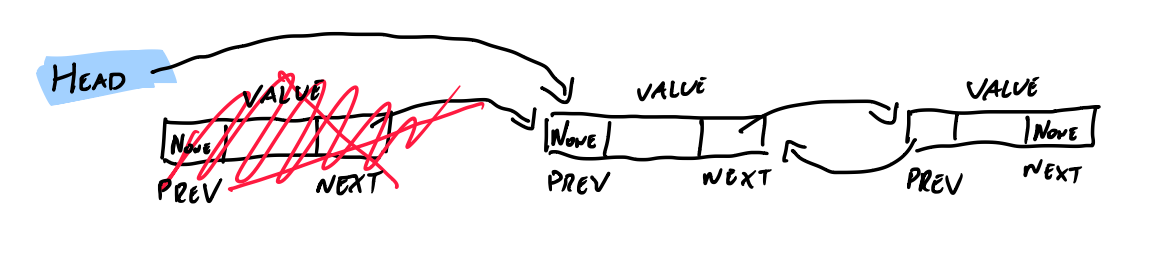
TO REMOVE THE FIRST NODE OF THE LIST



HEAD = HEAD.NEXT



HEAD.PREV = NONE



TO ADD AN ELEMENT TO THE LIST:

```
N = LIST()
N.PREV = NONE
N.NEXT = HEAD
HEAD.PREV = N
HEAD = N
```

} $O(1)$

THIS ALLOWS ME TO IMPLEMENT ① IN $O(1)$ TIME.

A RUN THROUGH THE BODY OF THE LOOP THEN COSTS $O(1)$ $(O(1) + O(1) + O(1) + O(1) \le O(1))$.

GIVEN THAT THE NUMBER OF ITERATIONS IS $\le n^2$, THE RUNTIME IS $O(n^2)$. (GIVEN THAT EACH OF THE 5 DATA STRUCTURES CAN BE INITIALIZED IN $O(n^2)$ TIME) (AND, ONCE AGAIN, $\underbrace{O(n^2) + O(n^2) + O(n^2) + O(n^2) + O(n^2)}_{5 \text{ TIMES}} \le O(n^2)$)