O, Ω, Θ NOTATIONS

TO AVOID GETTING STUCK INTO TOO MANY CONSTANTS, WE JUST DISREGARD THEM ALTOGETHER.

SUPPOSE THAT $T(n)$ IS THE RUNTIME OF AN ALGORITHM.

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

WE WRITE $T(n) = O(f(n))$ OR $T(n) \leq O(f(n))$

(O IS FOR UPPER BOUNDS)

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

(Ω IS FOR LOWER BOUNDS)

$1.1\, n^2 \stackrel{?}{=} O(n^3)$ ✓	$1.1\, n^2 \stackrel{?}{=} \Omega(n^3)$ ✗
$1.1\, n^2 \stackrel{?}{=} O(n^2)$ ✓	$1.1\, n^2 \stackrel{?}{=} \Omega(n^2)$ ✓
$1.1\, n^2 \stackrel{?}{=} O(n)$ ✗	$1.1\, n^2 \stackrel{?}{=} \Omega(n)$ ✓

DEF: IF $f(n) = O(T(n))$ AND $f(n) = \Omega(T(n))$, WE WRITE $f(n) = \Theta(T(n))$

DEF F(n):
RETURN n*n

- GALE-SHAPLEY():
  - INITIALLY, EACH $a \in A$ AND EACH $b \in B$ IS "FREE"
  - WHILE THERE IS SOME FREE $a_i \in A$ THAT HASN'T YET PROPOSED TO EACH $b \in B$: $\leq n^2$
    - LET a_i BE A FREE ELEMENT OF A. ①
    - LET $B' \subseteq B$ BE THE SET OF ELEMENTS OF B THAT a_i HASN'T PROPOSED TO YET.
    - LET $b_j \in B'$ BE THE ELEMENT OF B' THAT RANKS HIGHEST IN a_i'S PREFERENCE LIST. ②
    - IF b_j IS FREE: ③
      - MATCH a_i AND b_j [a_i AND b_j GET ENGAGED]
      - a_i AND b_j ARE NOT FREE ANYMORE
    - ELSE:
      - SUPPOSE THAT b_j IS MATCHED TO $a_k \in A$
      - IF b_j PREFERS a_k TO a_i: ④
        - a_i REMAINS FREE, AND $\{a_k, b_j\}$ REMAIN ENGAGED
      - ELSE:
        - "BREAK UP" $\{a_k, b_j\}$
        - MATCH $\{a_i, b_j\}$
        - a_k BECOMES FREE.

---

① IDENTIFY A FREE a_i

② IDENTIFY THE b_j THAT (i) RANKS HIGHEST IN a_i'S PREFERENCE ORDER, AND THAT (ii) a_i HASN'T YET PROPOSED TO.

③ DETERMINE WHETHER b_j IS FREE, AND – IF SHE'S NOT – FIND HER CURRENT PARTNER a_k

④ DETERMINE WHETHER b_j LIKES a_k MORE THAN a_i.

TO IMPLEMENT THESE OPERATIONS IN CONSTANT TIME, WE USE NON-TRIVIAL DATA STRUCTURES:

Ⓐ APREF, A $n \times n$ ARRAY, SUCH THAT APREF[i][ℓ] CONTAINS THE INDEX OF THE ℓ^{TH} MOST PREFERRED PARTNER OF a_i (ACCORDING TO a_i'S RANKING).

$a_3 : b_2 > b_1 > b_4 > b_3$

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 ORDERING, OF THE NEXT b_j THAT a_i IS GOING TO PROPOSE TO.

AT THE OUTSET, WE INITIALIZE NEXT[i] = 1 FOR EACH i.

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

AFTER a_i PROPOSES, WE SET NEXT[i] += 1.

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

Ⓒ CURRENT, AN ARRAY OF SIZE n, SUCH THAT CURRENT[i] CONTAINS THE j SUCH THAT b_i IS CURRENTLY ENGAGED TO a_j.

IF b_i IS CURRENTLY FREE, WE SET CURRENT[i] = NONE (CURRENT[i] = ⊥)

THUS, CURRENT IMPLEMENTS 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

$b_3 : a_2 > a_1 > a_3 > a_4$

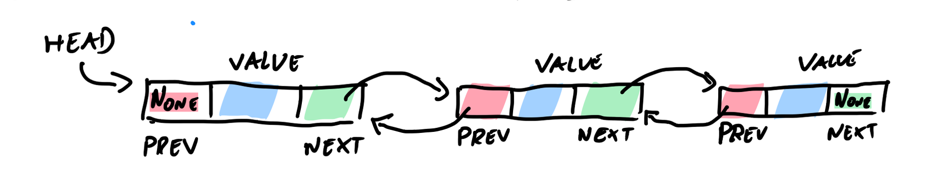
RANKING[3][2] = 1
RANKING[3][1] = 2
RANKING[3][3] = 3
RANKING[3][4] = 4

THUS, RANKING IMPLEMENTS ④ IN $O(1)$ TIME.

THUS, ②, ③ AND ④ CAN BE IMPLEMENTED IN $O(1)$ (CONSTANT) TIME.

WHAT ABOUT ① ?

WE USE A DOUBLY-LINKED LIST.



TO GET THE FIRST VALUE (THE VALUE OF THE FIRST ITEM) OF THE LIST, I CAN WRITE (IN PYTHON)

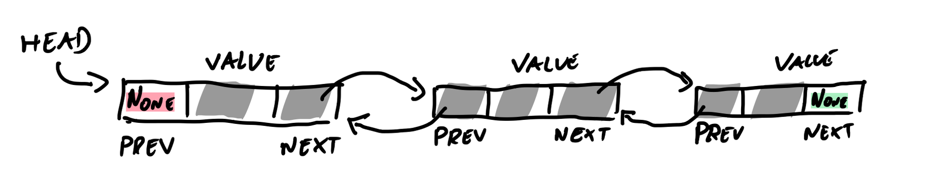
HEAD.VALUE

TO GET THE SECOND ITEM OF THE LIST, I CAN USE HEAD.NEXT (THE VALUE OF THE SECOND ITEM IS HEAD.NEXT.VALUE)

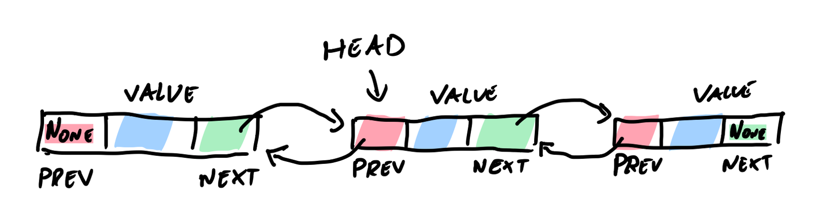
HEAD.NEXT.PREV.VALUE $\stackrel{?}{=}$ HEAD.VALUE

TO GET THE FIRST ELEMENT OF THE LIST WE USE HEAD (OR HEAD.VALUE IF WE WANT ITS VALUE).

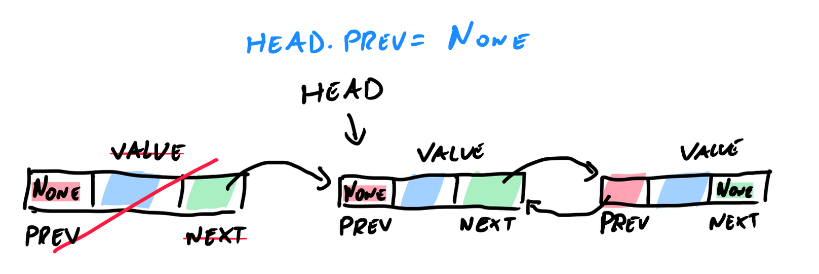
TO REMOVE THE FIRST ELEMENT OF THE LIST WE JUST RUN HEAD = HEAD.NEXT, HEAD.PREV = None



HEAD = HEAD.NEXT



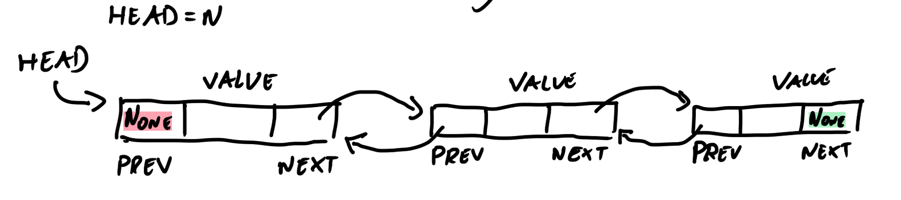
HEAD.PREV = None



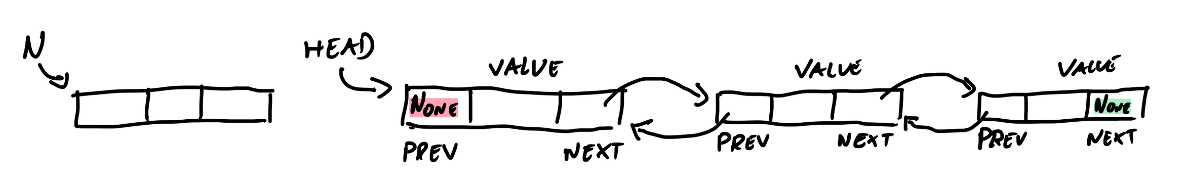
TO ADD AN ELEMENT TO THE LIST:

```
N = List()
N.PREV = None
N.NEXT = HEAD
HEAD.PREV = N
HEAD = N
```

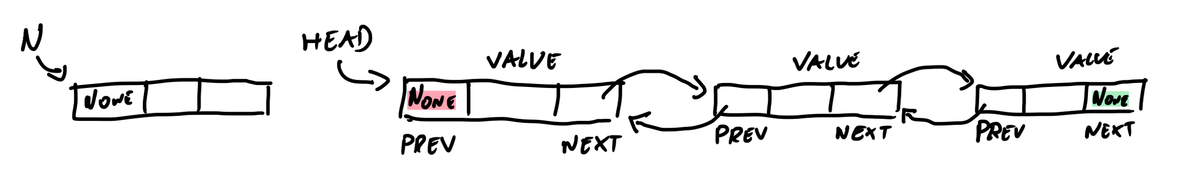
} $O(1)$



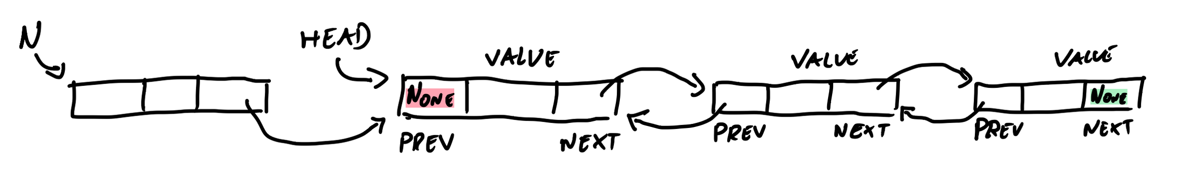
N = List()



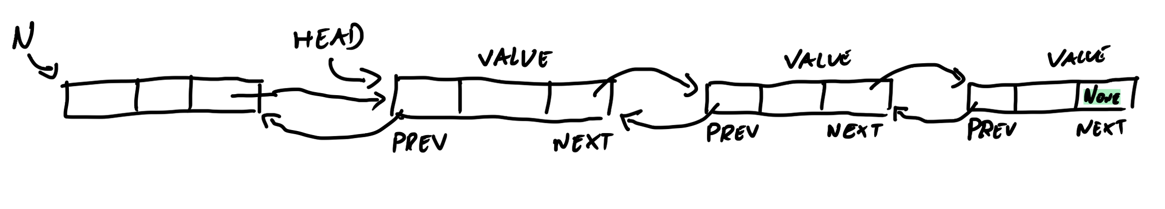
N.PREV = None



N.NEXT = HEAD



HEAD.PREV = N



HEAD = N

THUS, OPERATION ① CAN BE IMPLEMENTED IN $O(1)$ TIME.

A RUN THROUGH THE BODY OF THE LOOP THEN TAKES $O(1)$ TIME $\left(O(1) + O(1) + O(1) + O(1) \leq O(1) \right)$

SINCE THE LOOP RUNS FOR AT MOST $O(n^2)$ ITERATIONS, THE TOTAL RUNTIME IS AT MOST $O(n^2) \cdot O(1) = O(n^2)$.