26 AUGUST
SATURDAY

JULY 2023
M T W T F S S M T W T F S S M T W T F S S
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Google Online.

Every language has a function to generate a random number. How is this function made?

3 → weight. Higest weight → Highest probability.
5 → weight.
2 → weight.

528 Random Pick with Weight.

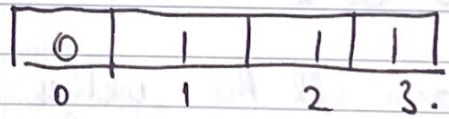
w[i] = weight of i^{th} index

w[i] = [1, 3]
0 1.

probability of 0^{th} index = $\frac{1}{4}$ = 25%.
probability of 1^{th} index = $\frac{3}{4}$. = 75%.

27 SUNDAY

Approach One.

E = 

rand(). an Index
Then return.
E[rand()].

2023

Prefix Sum Array = P.

$p[i] = p[0] .... + p[i]$

$W = [4, 7, 2, 5]$.

Sum (W) = 18.

$P = [4, 11, 13, 18]$

rand() % total

r = rand() % total $r \in (0, total - 1)$.

$r \Rightarrow [0, ... 17]$.

$P = [4, 11, 13, 18]$

0-3 4-10 11-12 13-17. To search r we can do <u>Linear Search</u>.

r = 5 so it lies in 4-10 -> index = 11.

Code :-

```
vector<int> prefix;
Solution(vector<int> &w) {
    int n = w.size();
    prefix.resize(n);
    prefix[0] = w[0];
    for(int j=0; j<n; i++) {
        prefix[i] = w[i] + prefix(i-1];
    }
}
```

Simple ⇒ linear Search.

p = [4, 11, 13, 18]
0-3 4-10 11-12 13-17.

To find the ranges in which r lies we need to find UpperBound
↳ First index that is greater than r.

So basically we apply binary search. instead of linear search. as we get time complexity O(logn) instead of O(n)

```
int pickIndex() {
    int total = prefix.back();
    ~~int rand~~ int r = rand() % total;

    int lo = 0; int hi = prefix.size() -1;
    int ans = 0;
    while (lo <= hi) {
        int mid = (lo + hi)/2;
        if (prefix[mid] <= r) {
            lo = mid + 1; // go to right side for UB
        }
        else {
            ans = mid;
            hi = mid - 1;
        }
    }
    return ans;
}
```