26 AUGUST

SATURDAY

Google Online.

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

9 3 → weight. Higest weight → Highest probablity.

10 5 → weight.

2 → weight.

11

<u>528</u> Random Pick with Weight.

12 $w[i]$ = weight of i^{th} index

1 $w[i] = [1, 3]$

2 0 1.

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

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

27 SUNDAY Approach One.

E =

0	1	1	1
0	1	2	3.

rand(). an index

Then return.

E[rand()].

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 ∈ (0, total -1).

r => [0, ... 17].

p = [4, 11, 13, 18]
0-3 4-10 11-12 13-17.   To search r we can do Linear Search

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 range 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) of 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 ↑val
            }
            else {
              ans = mid;
              hi = mid -1;
            }
    }
}   return ans;.
```