❮ Previous Next ❯

497. Random Point in Non-overlapping Rectangles

July 26, 2018 | 5.4K views

★★★★★
Average Rating: 3.44 (9 votes)

Given a list of **non-overlapping** axis-aligned rectangles `rects`, write a function `pick` which randomly and uniformily picks an **integer point** in the space covered by the rectangles.

Note:

1. An **integer point** is a point that has integer coordinates.
2. A point on the perimeter of a rectangle is **included** in the space covered by the rectangles.
3. `i` th rectangle = `rects[i]` = `[x1,y1,x2,y2]`, where `[x1, y1]` are the integer coordinates of the bottom-left corner, and `[x2, y2]` are the integer coordinates of the top-right corner.
4. length and width of each rectangle does not exceed `2000`.
5. `1 <= rects.length <= 100`
6. `pick` return a point as an array of integer coordinates `[p_x, p_y]`
7. `pick` is called at most `10000` times.

Example 1:

```
Input:
["Solution","pick","pick","pick"]
[[[[1,1,5,5]]],[],[],[]]
Output:
[null,[4,1],[4,1],[3,3]]
```

Example 2:

```
Input:
["Solution","pick","pick","pick","pick","pick"]
[[[[-2,-2,-1,-1],[1,0,3,0]]],[],[],[],[],[]]
Output:
[null,[-1,-2],[2,0],[-2,-1],[3,0],[-2,-2]]
```

Explanation of Input Syntax:

The input is two lists: the subroutines called and their arguments. `Solution`'s constructor has one argument, the array of rectangles `rects`. `pick` has no arguments. Arguments are always wrapped with a list, even if there aren't any.

Solution

Approach 1: Prefix Sum and Binary Search

Intuition

Some rectangles may be more likely to be sampled from than others, since some may contain more points than others, and each point has an equal chance of being sampled. Is there a way to select a rectangle to sample from, such that the probabilities are proportional to the number of points contained in each rectangle? Is there a way to do this using less than $O(\text{total number of points})$ space?

Algorithm

Create a weight array w, where $w[i]$ is the number of points in $\text{rects}[i]$.

Let $\textbf{tot} = \sum_{i=0}^{N-1} w[i]$, where $N = \text{len}(w)$.

Compute the prefix sum array p, where $p[x] = \sum_{i=0}^{x} w[i]$.

Generate a random integer targ in the range $[0, \text{tot})$.

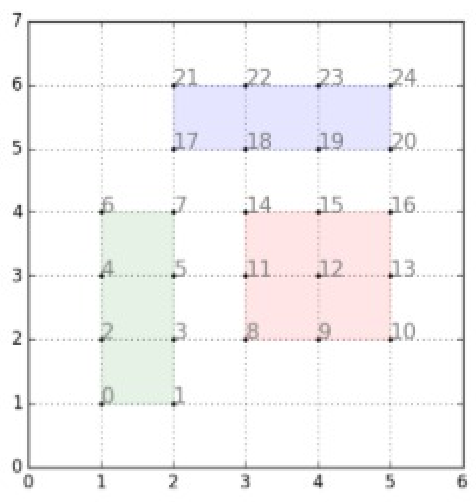
Use binary search to find the index x where x is the lowest index such that $\text{targ} < p[x]$. $\text{rects}[x]$ is the rectangle that we will sample from.

Note that for some index i, all integers v where $p[i] - w[i] \leq v < p[i]$ map to this index. Therefore, rectangles will be sampled proportionally to the rectangle weights.

The only step remaining is to choose a random point in $\text{rects}[x]$. Generating random $x\_coordinate$ and $y\_coordinate$ within this rectangle area will suffice, but we can also reuse targ by mapping it to the point

$$x\_coordinate = x1 + (targ - p[i] + w[i]) \ \% \ (x2 - x1 + 1)$$
$$y\_coordinate = y1 + (targ - p[i] + w[i]) \ / \ (x2 - x1 + 1)$$

This strategy is useful when calls to the random number generator are expensive.



Mapping from targ to x_coordinate and y_coordinate for rects = [[1, 1, 2, 4], [3, 2, 5, 4], [2, 5, 5, 6]]

C++ Java Copy

```
class Solution {
public:

    vector<vector<int>> rects;
    vector<int> psum;
    int tot = 0;
    //c++11 random integer generation
    mt19937 rng{random_device{}()};
    uniform_int_distribution<int> uni;

    Solution(vector<vector<int>> rects) {
        this->rects = rects;
        for (auto& x : rects) {
            tot += (x[2] - x[0] + 1) * (x[3] - x[1] + 1);
            psum.push_back(tot);
        }
        uni = uniform_int_distribution<int>{0, tot - 1};
    }

    vector<int> pick() {
        int targ = uni(rng);

        int lo = 0;
        int hi = rects.size() - 1;
        while (lo != hi) {
            int mid = (lo + hi) / 2;
            if (targ >= psum[mid]) lo = mid + 1;
```

Complexity Analysis

- Time Complexity: $O(N)$ preprocessing. $O(\log(N))$ pick.
- Space Complexity: $O(N)$

Rate this article: ★★★★★

❮ Previous Next ❯

Comments: 3 Sort By ▾

Type comment here... (Markdown is supported)

Preview Post

feng-zhe ★ 83 July 31, 2018 12:42 AM

Will this trick work?
`x_coordinate=x1+(targ-p[i]+w[i]) % (x2-x1+1)`
`y_coordinate=y1+(targ-p[i]+w[i]) / (x2-x1+1)`
Because I feel that the x and y of the randomly picked point should be independent. But by this trick, it seems it introduces unnecessary relationships between x and y.

Read More

3 ⌃ ⌄ | Share | Reply

SHOW 1 REPLY

sreejoy ★ 16 February 24, 2019 10:56 AM

Please fix the notations "Let \text{tot} = \sum_\limits{i=0}^{N-1}w[i], where N=len(w)N = \text{len}(w)N=len(w)."

2 ⌃ ⌄ | Share | Reply

david2999999 ★ 48 June 13, 2020 3:47 PM

```
class Solution {
    private static final int X1 = 0;
    private static final int Y1 = 1;
    private static final int X2 = 2;
```

Read More

0 ⌃ ⌄ | Share | Reply