# 6117. The Latest Time to Catch a Bus

My Submissions (/contest/biweekly-contest-82/problems/the-latest-time-to-catch-a-bus/submissions/) Back to Contest (/contest/biweekly-contest-82/)

| | |
|---|---|
| User Accepted: | 0 |
| User Tried: | 0 |
| Total Accepted: | 0 |
| Total Submissions: | 0 |
| Difficulty: | Medium |

You are given a **0-indexed** integer array `buses` of length `n`, where `buses[i]` represents the departure time of the $i^{th}$ bus. You are also given a **0-indexed** integer array `passengers` of length `m`, where `passengers[j]` represents the arrival time of the $j^{th}$ passenger. All bus departure times are unique. All passenger arrival times are unique.

You are given an integer `capacity`, which represents the **maximum** number of passengers that can get on each bus.

The passengers will get on the next available bus. You can get on a bus that will depart at `x` minutes if you arrive at `y` minutes where `y <= x`, and the bus is not full. Passengers with the **earliest** arrival times get on the bus first.

Return *the latest time you may arrive at the bus station to catch a bus*. You **cannot** arrive at the same time as another passenger.

**Note:** The arrays `buses` and `passengers` are not necessarily sorted.

**Example 1:**

```
Input: buses = [10,20], passengers = [2,17,18,19], capacity = 2
Output: 16
Explanation:
The 1st bus departs with the 1st passenger.
The 2nd bus departs with you and the 2nd passenger.
Note that you must not arrive at the same time as the passengers, which is why you must arrive before the 2nd passenger to catch
```

**Example 2:**

```
Input: buses = [20,30,10], passengers = [19,13,26,4,25,11,21], capacity = 2
Output: 20
Explanation:
The 1st bus departs with the 4th passenger.
The 2nd bus departs with the 6th and 2nd passengers.
The 3rd bus departs with the 1st passenger and you.
```

**Constraints:**

- `n == buses.length`
- `m == passengers.length`
- `1 <= n, m, capacity <= 10^5`
- `2 <= buses[i], passengers[i] <= 10^9`
- Each element in `buses` is **unique**.
- Each element in `passengers` is **unique**.

JavaScript

```javascript
function Bisect() {
    return { insort_right, insort_left, bisect_left, bisect_right }
    function insort_right(a, x, lo = 0, hi = null) {
        lo = bisect_right(a, x, lo, hi);
        a.splice(lo, 0, x);
    }
    function bisect_right(a, x, lo = 0, hi = null) { // > upper_bound
        if (lo < 0) throw new Error('lo must be non-negative');
        if (hi == null) hi = a.length;
        while (lo < hi) {
            let mid = parseInt((lo + hi) / 2);
            a[mid] > x ? hi = mid : lo = mid + 1;
        }
        return lo;
    }
    function insort_left(a, x, lo = 0, hi = null) {
        lo = bisect_left(a, x, lo, hi);
        a.splice(lo, 0, x);
```

```
    }
    function bisect_left(a, x, lo = 0, hi = null) { // >= lower_bound
        if (lo < 0) throw new Error('lo must be non-negative');
        if (hi == null) hi = a.length;
        while (lo < hi) {
            let mid = parseInt((lo + hi) / 2);
            a[mid] < x ? lo = mid + 1 : hi = mid;
        }
        return lo;
    }
}


const sm = (a) => a.reduce(((x, y) => x + y), 0);
const sumOfRange = (l, r) => (l + r) * (r - l + 1) / 2;
const isRangeOfStep = (a) => sm(a) == sumOfRange(a[0], a[a.length - 1]);
const stmkey_in = (m) => new Map([...m].sort((x, y) => x[0] - y[0]));
const generateMapSet = (m, k, v, cap) => {
    if (!m.has(k)) m.set(k, new Set());
    if (m.get(k).size < cap) {
        m.get(k).add(v);
        return true;
    }
    return false;
};

/*
[
  2,  3,  5,  7,  8,
  9, 12, 13, 18, 20
]
[
   2,  4,  5,  8, 10,
  12, 13, 14, 18, 19,
  30, 34
]

Map {
  2 => Set { 2 },
  5 => Set { 4 },
  7 => Set { 5 },
  8 => Set { 8 },
  12 => Set { 10 },
  13 => Set { 12 },
  18 => Set { 13 },
  20 => Set { 14 }
}
*/
const latestTimeCatchTheBus = (a, b, capacity) => {
    let m = new Map(), bi = new Bisect();
    a.sort((x, y) => x - y);
    b.sort((x, y) => x - y);
    // pr(a);
    // pr(b);
    for (const x of b) {
        // pr("after", m)
        let idx = bi.bisect_left(a, x);
        while ((m.get(a[idx]) || new Set()).size >= capacity) idx++;
        if (a[idx] != undefined) {
            // pr(x, "bus", a[idx])
            let op = generateMapSet(m, a[idx], x, capacity);
            if (!op) {
                while ((m.get(a[idx]) || new Set()).size >= capacity) idx++;
                // pr(x, "bus", a[idx])
                if (a[idx] != undefined) generateMapSet(m, a[idx], x, capacity);
            }
        }
    }
    for (const bus of a) { // add empty bus
        if (!m.has(bus)) m.set(bus, new Set())
    }
    m = stmkey_in(m);
    // pr(m, "cap", capacity);
    let buses = Array.from(m.keys()).reverse(), used = new Set(b);
    // pr(buses)
    for (const bus of buses) {
        let se = m.get(bus), d = [...se], first = d[0], last = d[d.length - 1];
```

```
        if (d.length == 0) return bus;
        if (isRangeOfStep(d)) {
            if (last < bus && d.length < capacity) {
                // pr("111")
                res = Math.max(bus, last + 1);
                if (!used.has(res)) return res;
            } else {
                // pr("222")
                res = first - 1;
                if (!used.has(res)) return res;
            }
        } else {
            if (last < bus && d.length < capacity) {
                // pr("333")
                res = Math.max(bus, last + 1);
                if (!used.has(res)) return res;
            } else {
                // pr("444")
                for (let i = d.length - 1, expect = last; ~i; i--, expect--) {
                    if (d[i] != expect && !used.has(expect)) return expect;
                }
            }
        }
    }
    return a[0]
};
```

Custom Testcase Use Example Testcases

Run Submit

Submission Result: Accepted (/submissions/detail/742777449/) More Details > (/submissions/detail/742777449/)

Share your acceptance!

1