07-C1

搜索树应用

范围查询：一维

邓俊辉

deng@tsinghua.edu.cn

你这个人太敏感了。这个社会什么都需要，唯独不需要敏感

1D Range Query

❖ 考查x轴上的n个点：$P = \{ p_1, p_2, p_3, \ldots, p_n \}$

❖ 任给区间 $I = (x_1, x_2]$

- 计数/COUNTING：P 中有多少点落在其中?
- 报告/REPORTING：列出 $I \cap P$ 中的所有点

❖ [Online] 通常 P 相对固定，而 I 是不断随机更新的

❖ 如何将 P 预处理为适当的数据结构，使得每次查询都能高效地完成?

Brute-Force

❖ 逐个地取出P中的点p，在$\mathcal{O}(1)$时间内判断是否$p \in (x_1, x_2]$

如此，可以在线性时间内完成每次查询

❖ 可以做得更快吗？貌似不可能，毕竟...

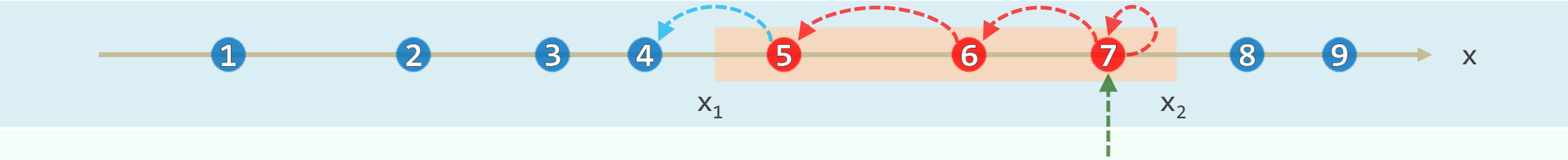
❖ 在最坏情况下会有$\Omega(n)$个点命中

即便是列举出它们，也需要$\Omega(n)$时间

❖ 然而，要是暂且不计列举输出的环节，仅考查搜索的环节呢?

Binary Search

❖ 将所有点预处理为一个有序向量（并添加哨兵$p_0 = -\infty$）



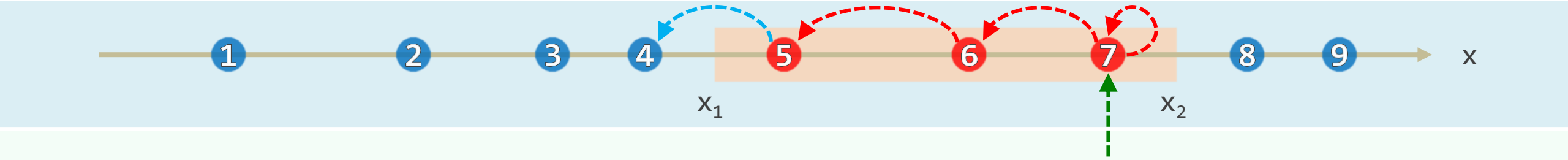
❖ 任给区间 $I = (x_1, x_2]$

- 通过二分查找确定 t = search(x_2) = max{ i | $p_i \le x_2$ } //$\mathcal{O}$(logn)
- 从p_t出发，自后向前遍历向量，直到在p_s处越出查询区间I //$\mathcal{O}$(r)

  在(s, t]内的每一个点，都予输出 //它们恰好就是所有的输出
- 返回命中计数 r = t - s //output size

Output-Sensitivity

❖ 如此，每次查询的结果，都可在 $\mathcal{O}(1 + r + \log n)$ 时间内报告出来



❖ 自然，借助二分查找，p_s 也可 $\mathcal{O}(\log n)$ 时间内确定

于是，如果仅仅着眼于**计数**查询，每次只需 $\mathcal{O}(\log n)$ 时间　//与r无关

❖ 关键是，上述策略**如何**推广到**二维**的范围查询？

甚至于，**可否**推广？

很遗憾，据我所知，还不能！