CMSC 420: Spring 2022

Homework 4: Range Trees, Hashing, and Tries

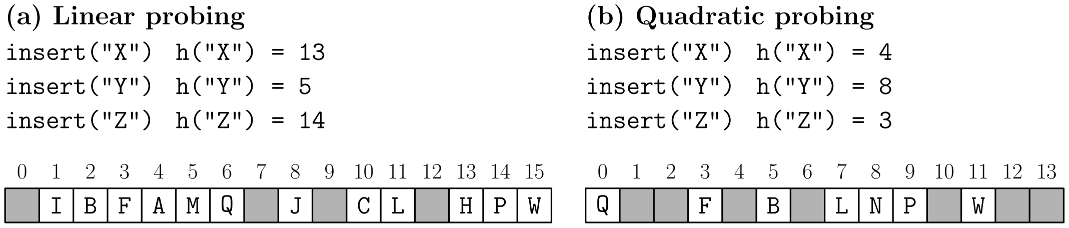
Handed out Tue, May 3. Due **Tue, May 10, 11:00am**. (Solutions will be discussed in class on Tue, May 10, so turn in whatever you have finished by then.) You may drop the lowest of your four homework scores. Even if you do not attempt this assignment, note that the material will be covered on the final exam.

Problem 1. (15 points) In this problem, you will show the result of inserting a sequence of three keys into a hash table, using linear and quadratic probing and double hashing. In each case, at a minimum indicate the following:

- Was the insertion successful? (The insertion fails if the probe sequence loops infinitely without finding an empty slot.)
- If the insertion is successful, indicate the number of *probes*, that is, the number of array elements accessed. (The initial access counts as a probe, so if there is no collision, the number of probes is 1.)
- Show contents of the hash table after each insertion. (You will show three tables for each part.)

For the purposes of assigning partial credit, you can illustrate the probes made as we did in the lecture notes (with little arrows).

(a) (5 points) Show the results of inserting the keys "X" then "Y" then "Z" into the hash table shown in Fig. 1(a), assuming *linear probing*. (Insert the keys in sequence, so if all are successful, the final table will contain all three keys.)



(a) Linear probing

```
insert("X")  h("X") = 13
insert("Y")  h("Y") = 5
insert("Z")  h("Z") = 14
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
▨	I	B	F	A	M	Q	▨	J	▨	C	L	▨	H	P	W

(b) Quadratic probing

```
insert("X")  h("X") = 4
insert("Y")  h("Y") = 8
insert("Z")  h("Z") = 3
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13
Q	▨	▨	F	▨	B	▨	L	N	P	▨	W	▨	▨

Figure 1: Hashing with linear and quadratic probing.

(b) (5 points) Repeat (a) using the hash table shown in Fig. 1(b) assuming *quadratic probing*.

(c) (5 points) Repeat (a) using the hash table shown in Fig. 2 assuming *double hashing*, where the second hash function g is shown in the figure.

Problem 2. (15 points) In this problem we will build a suffix tree for $S =$ `"baabaababaa$"`.

(a) (7 points) Recall that the 12 suffixes of S are (in reverse order):

$$S_{11} = \texttt{"\$"}, \quad S_{10} = \texttt{"a\$"}, \quad S_9 = \texttt{"aa\$"}, \quad \ldots, \quad S_0 = \texttt{"baabaababaa\$"}.$$

(c) Double hashing

```
insert("X")  h("X") = 13; g("X") = 6
insert("Y")  h("Y") = 4; g("Y") = 3
insert("Z")  h("Z") = 3; g("Z") = 4
```

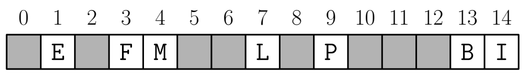


Figure 2: Hashing with double hashing.

Let id_j denote the *substring identifier* for S_j. (Recall from Lecture 17 that this is defined to be the shortest prefix of S_j that uniquely identifies it.) List all 12 substring identifiers for these suffixes in index order (from first to last $\mathrm{id}_0 \ldots \mathrm{id}_{11}$).

(b) (8 points) Draw the suffix tree for S. Draw your tree in the same edge labeling style we used in Fig. 7 in Lecture 17 LaTeX lecture notes. Order the children of each node in alphabetical order from left to right. (The form of your drawing is important. There are many online suffix-tree generators, and if it appears that you copied your answer from one of these, you will receive no credit.)

Hint: Begin by writing out all the substring identifiers in alphabetical order, one above the other. This makes it easy to determine common substrings.

Problem 3. (14 points) In this problem, we will consider how to use/modify range trees to answer two queries efficiently. Throughout, $P = \{p_1, \ldots, p_n\}$ is a set of n points in $\mathbb{R}^2$ (Fig. 3(a)). Your answer should be based on range trees, you may make modifications to P including possibly transforming the points and adding additional coordinates.

In each case, the various layers of your search structure (what points are stored there and how they are sorted) and explain how your search algorithm operates. An English explanation (as opposed to pseudocode) is sufficient. Justify your algorithm's correctness and derive its running time.

(a) (7 points) Assume that all the points of P have positive x- and y-coordinates. In a *platform-dropping query*, we are given a point $q = (q_x, q_y)$ with positive coordinates. This defines a horizontal segment running from the y-axis to q. The objective is to report the first point $p \in P$ that would be hit if we drop the platform (see Fig. 3(b)). Formally, this point has the maximum y-coordinate such that $p_x \leq q_x$ and $p_y \leq q_y$. If no point of P is hit by the platform, the query returns `null`.

Hint: Your data structure should use $O(n \log n)$ storage and answer queries in $O(\log^2 n)$ time.

(b) (7 points) In a *max empty-triangle query* you are given a point $q = (q_x, q_y)$. The objective is to compute the largest axis-parallel 45-45 right triangle that extends to the upper-right of q and contains no point of P in its interior. The answer to the query is the point of P that lies on the triangle's hypotenuse (see Fig. 3(c)). (Alternatively, you can think of this as sliding the 45° hypotenuse until it first hits a point of P). If the triangle can be grown to infinite size, return `null`.

Hint: Your data structure should use $O(n \log^2 n)$ storage and answer queries in $O(\log^3 n)$ time.

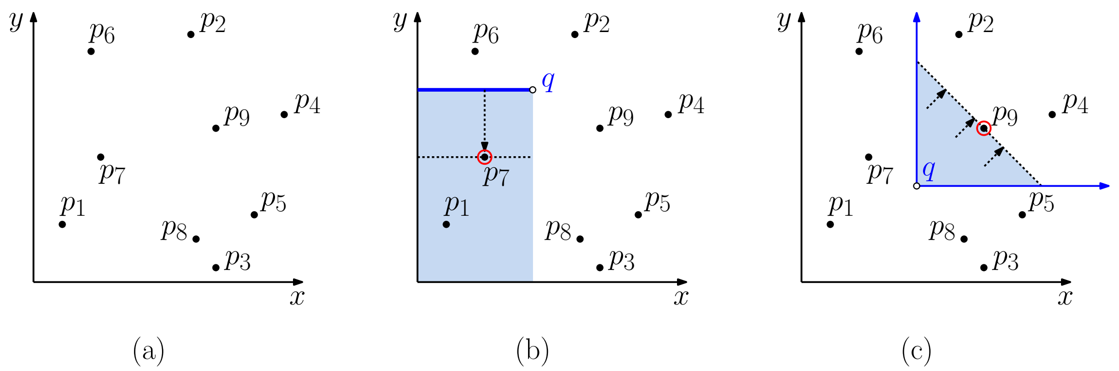


Figure 3: Platform-dropping and max empty-triangle queries.

Problem 4. (6 points) Consider the buddy-system memory allocation shown in Fig. 4, where shaded blocks are allocated and white blocks are free. Suppose that we deallocate the block of size 1 at address 22. Explain which blocks are merged together, and what single block replaces them all.

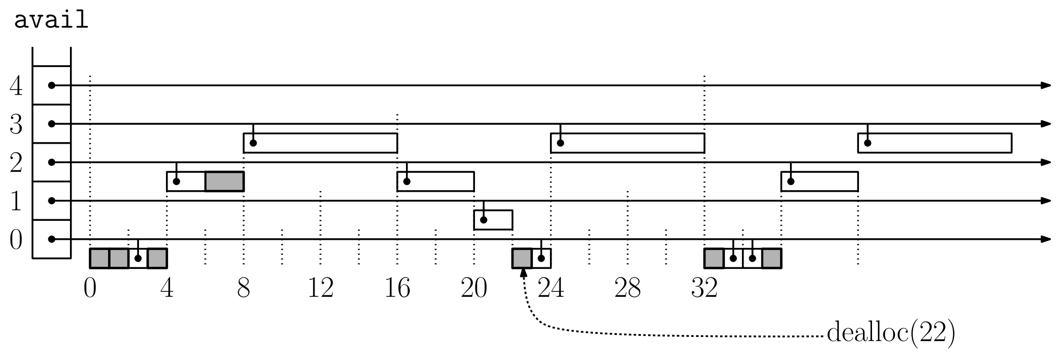


Figure 4: Buddy system memory allocation.