The second term is the size of the buffer that is allocated to read posting lists. We use the longest posting list read to calculate the size. Since `MergeList-D` only needs the set IDs in the each posting list, and does not need the token positions and set sizes, the buffer size is simply the total size of integer set IDs in the longest posting list.

$$|W_{ML}| \cdot 2 \cdot \textsc{SizeOf}(Int) + \max_{i \in [1, |Q|]} f_i \cdot \textsc{SizeOf}(Int) \quad (16)$$

We use Equation 17 to calculate the memory footprint of `ProbeSet-D`. The first term is the size of the buffer allocated for reading posting lists. Due to the use of prefix filter (subset of posting lists from 1 to p^*), the buffer size is the size of the longest posting list in the prefix. However, since `ProbeSet-D` needs the token positions and set sizes for position filter, in addition to set IDs, the size of each posting list entry is 3 integers. The second term is the size of the buffer allocated for reading sets. Similar to the other buffer, its size is calculated using the maximum size read. The third term is the size of the hash set allocated for tracking sets that have been pruned or read, so it is simply the size of the IDs of all the sets that appeared in the prefix posting lists. The set of all sets appeared in the prefix is W.

$$\begin{aligned} \max_{i \in [1, p^*]} 3 f_i \cdot \textsc{SizeOf}(Int) + \max_{X \in W \setminus V} |X[j_{X,0} :]| \\ + |W| \cdot \textsc{SizeOf}(Int) \end{aligned} \quad (17)$$

Equation 18 is used to calculate the memory footprint for `JOSIE-D`, where δ is the extra posting lists read after the prefix filter, V^* is the set of pruned candidates using position filter, and W_i is the set of candidates after reading posting list i. See Equation 14 and 7 for their usages in running time analysis. The first three terms have nearly indentical expression as those of `ProbeSet-D`, however the magnitudes can be different, as $p^* + \delta \geq p^*$, and $W \setminus V^* \subseteq W \setminus V$. The last term is the maximum size of the hash map allocated for unread candidates (W_i) after reading a batch of posting lists ends at i. As described in Section 3.2, in addition to the set ID, we keep track of 4 additional integers for each candidate. Also, since pruned candidates are removed from the hash map, we use the maximum count of candidate sets during query processing to calculate the allocated size, as the deleted slots can be reused.

$$\begin{aligned} \max_{i \in [1, p^* + \delta]} 3 f_i \cdot \textsc{SizeOf}(Int) + \max_{X \in W \setminus V^*} |X[j_{X,0} :]| \\ + |W| \cdot \textsc{SizeOf}(Int) + \max_{i \in [1, p^* + \delta]} 5 |W_i| \cdot \textsc{SizeOf}(Int) \end{aligned} \quad (18)$$

For the LSH Ensemble algorithms, `LSHEnsemble-60` and `LSHEnsemble-90`, we use Equation 19. The first term is size of buffer allocated for reading sets, which is the maximum size over all sets read. The second term is the size of the hash set allocated for tracking the candidates retrieved from the LSH indexes.

$$\max_{X \in W_{LSH}} |X| + |W_{LSH}| \cdot \textsc{SizeOf}(Int) \quad (19)$$

During experiment, we record all the relevant variables needed to calculate memory footprint. The experimental results on Open Data and Web Table benchmarks are shown in Figure 5, 7, and 9. Please see Section 4 for detailed discussion.

7.4 Caching

The search algorithm itself does not deal with memory management or caching, which is handled by the storage layer which stores the posting lists and integer sets. The storage layer could be a separate process and its communications with the query prcessing process may involve serialization.

During the experiment, we used Postgres as the storage layer for posting lists and sets. In order to better reflect the worst case running time of the algorithms, we set the shared memory buffer size of Postgres to 128 MB, which is small enough compared to the data size (see Table 4) to avoid caching too many results.

In a practical deployment of the search engine, increasing the shared memory buffer size to utilize caching may be helpful, as it also depends on the choice of the storage layer, for example, a disk-based storage system would benefit but not an in-memory database.