CSCI 361 Week 11 Notes

Kyle Krstulich

April 1, 2025

Announcements

There is a guest lecturer today and we will cover push/pop after.

Hash Tables: Rehashed

Jacob Downs is our presenter. Montana native Montana grad. Hash tables take advantage of direct addressing. Direct addressing requires us to have precomputed values. And only supports unsigned integers. Its also $O(n)$ in memory, inefficient.

Hash functions allows us to have $O(m)$ memory allocations. m is usually prime, this helps in avoiding collisions, makes a more evenly distributed set.

Example of string hashing

$$\begin{aligned}\text{hash}(s) &= s[0] + s[1] \cdot p + s[2] \cdot p^2 + \cdots + s[n-1] \cdot p^{n-1} \\ &= \left(\sum_{i=0}^{n-1} s[i] \cdot p^i \right) \bmod m\end{aligned}$$

A good hash function is deterministic, fast, and handles collisions. However, integer based hash function will inevitably run into collisions with a large input n. To solve this we can linearly probe the next available space. This approach makes set and get run in $O(m)$.

You can also chain the values, instead of storing the value itself in the table, it stores a pointer to a linked list, or bucket. This allows us to use hash tables for more complex values. This segments our memory into a non contiguous memory. Drawbacks of chaining, you need additional memory to store linked lists, need to traverse list in potentially $O(n)$ time.

Let a hash table, H, store n items with m keys with a $\frac{n}{m}$ load factor, the average amount of entries in each table slot.. Its a metric to tell how full the hash table is. When this load factor is exceeded, the table starts becoming slow. A strategy for this is when the load factor ≥ 1 then to increase the size of the array by some geometric factor. Adding a fixed number of new elements is not a good strategy.

Double hashing is another solution, where you have two hash functions.

Example

Suppose we start with a hash table of size m. If the load factor $\frac{n}{m} > 1$ then we double the size of the array. Copy and rehash each element in previous array, do this 3 times, with a total amount of hashing be $m + 2m + 4m + 8m$. $M = 8m$ and $m = \frac{M}{8}$. With $\frac{M}{8} + \frac{M}{4} + \frac{M}{2} + \frac{M}{1} < 2M$.

Give feedback through discord or QR codes.

Homework Presentation

```
 def getPopD():
    return "@SP,AM=M-1,D=M,"


def getPushD():
    return "@SP,A=M,M=D,@SP,M=M+1,"
```

Assignment Details

We define push/pop, seg, index as

```
def f(push/pop, seg, index):
  if seg in ['local', 'argument', 'this', 'that']
  memory_address = LCL + index
```