Hashing

- Read section 7.3 (pages 269-275)
- What abstract data type are **hash tables** used for? What operations does that type support?
- What do we call the **hash function**? What is the **hash address** of a key?
- If our keys are characters, and we want to store them in a hash table of size $m = 10$, based on a hash function that counts the letters from 0 and then computes that number modulo m. What hash addresses would the letters in EXAMPLE correspond to?
- What are some important properties of a *good* hash function?
- What do we call a **collision** in a hash table?
- How does the **open hashing**, or **separate chaining** mechanism store keys?
  - How do we perform search/insertion/deletion in the open hashing case?
  - What factors influence the length of the resulting linked lists?
  - What number is the **load factor** of a hash table?
  - If we want to store 100 keys in a hash table of size 20, what is the load factor?
  - In a hash table with load factor α, how many elements need to be inspected on average, for a successful search? What about unsuccessful searches?
  - What consequence does having a large load factor (much less than 1) have?
  - What consequence does having a small load factor (much more than 1) have?
- In what way is the hash table a space-time trade-off?
- How does the **closed hashing**, or **open addressing** mechanism store keys?
  - What load factors can we have in this scheme?
  - How does **linear probing** resolve resolutions?
  - How does search work in this scheme? What about deletion?
  - What are **clusters** and why do they cause problems for closed hashing?
  - What is the average access time for closed hashing in terms of the load factor α, for both successful and unsuccessful searches?
  - What values of the load factor should we aim for in closed hashing?
- Practice problems: 7.3.1, 7.3.2, 7.3.3, 7.3.5, 7.3.7