COMPUTER ENGINEERING 12
PROJECT 5

Xiang Li
xli8@scu.edu

Outline

- Priority Queue (ADT)
- A sorting based on heapsort (testing program given to you) Week 1

- Huffman coding (testing program written by you) Week 2

Priority Queue

In a priority queue, an element with high priority is served before an element with low priority. If two elements have the same priority, they are served according to their order in the queue.

-- from WikiPedia

Priority Queue & Queue

In a queue, all the keys are ordered only according to when they enter the queue. Such order is not related to their priorities (e.g. values).

In a priority queue, both the key priorities (e.g. values) and their order of entering the queue are considered. In addition, priority plays a more important role.

Priority Queue Example

A priority queue example: Emergency room

Priority Queue Implementation

Assume you are required to organize a sequence as 5, 20, 18, 10, 3, 18, 20 in a priority queue. What is the dequeue sequence? (i.e. lower value indicating higher priority)

- How to implement a priority queue?
  - Can we use sorted array? What are the worst-case big-O for enqueue and dequeue?
  - Can we use sorted linked list? What are the worst-case big-O for enqueue and dequeue?
  - Can we do better?
    - Binary Heap

Week 1

Implementing a priority queue through a binary heap.

- PQ *createQueue(int (*compare)());

return a pointer to a new priority queue using compare as its comparison function

- void destroyQueue(PQ *pq);

deallocate memory associated with the priority queue pointed to by pq

- int numEntries(PQ *pq);

return the number of entries in the priority queue pointed to by pq

- void addEntry(PQ *pq, void *entry);

add entry to the priority queue pointed to by pq

- void *removeEntry(PQ *pq);

remove and return the smallest entry from the priority queue pointed to by pq

Priority Queue Struct

```
struct pqueue {
    int count;          /* number of entries in array */
    int length;                 /* length of allocated array  */
    void **data;                /* allocated array of entries */
    int (*compare)();           /* comparison function        */
};
```

Week 1

- void addEntry(PQ *pq, void *entry);

  step 1: check if the priority queue is full, if yes -> reallocate

  step 2: place the new element at the end of the binary heap

  step 3: reheap up.

- void *removeEntry(PQ *pq);

  step 1: remember the root

  Step 2: replace the root by the last element in the binary heap

  Step 3: reheap down.

Additional Notes

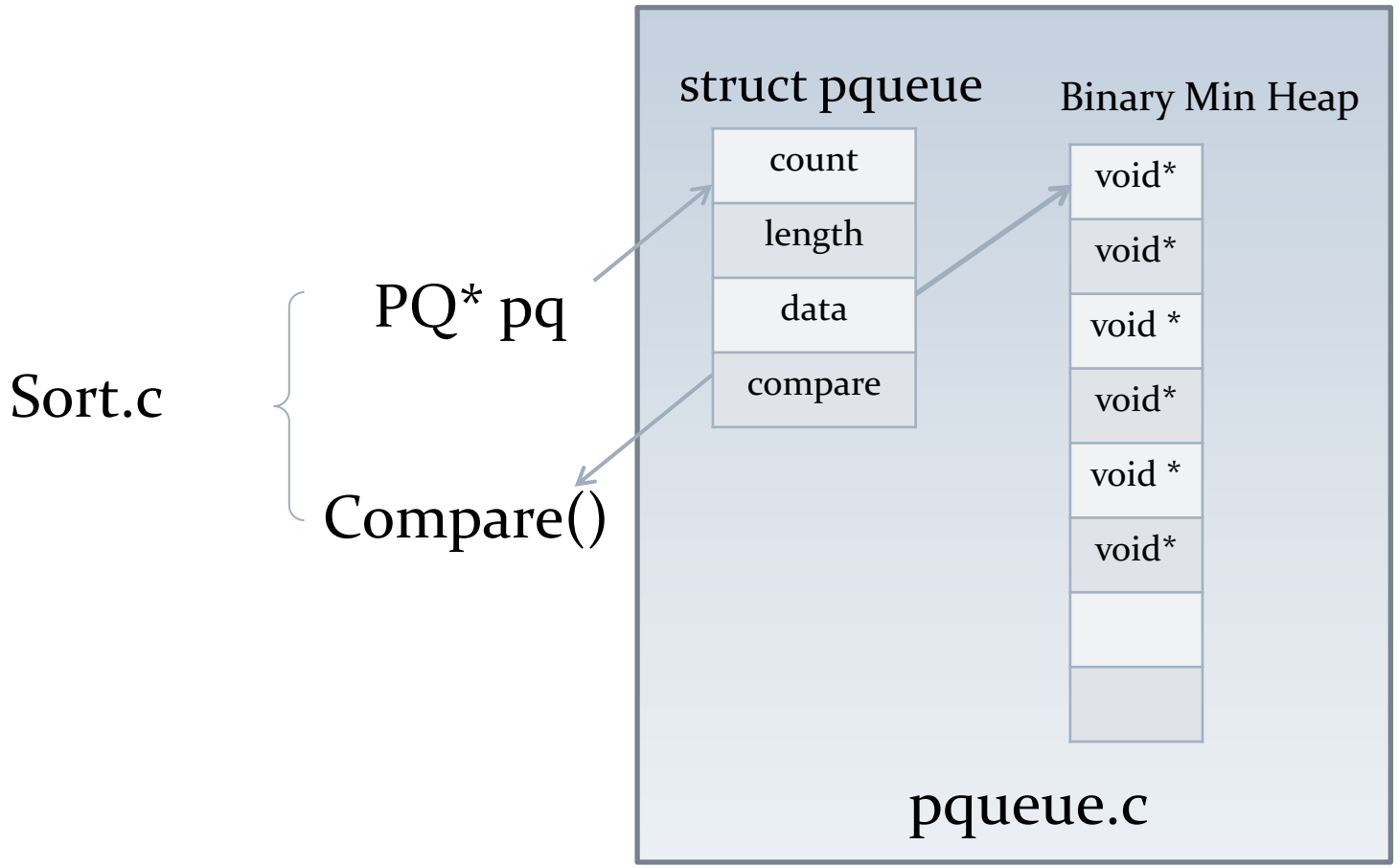
- Unlike previous data types that used arrays, the createQueue function does not require the maximum number of values as a parameter.
- Instead, Professor Loony wants you to increase the length of your array when it reaches capacity so it dynamically increases to accommodate the number of values. Thus, we eliminate the need for the client to inform us how large of an array it requires.

A sorting based on heapsort

- How it works?
  - Step 1. insert each input key into the binary heap.
  - Step 2. using a loop, where each iteration prints out the root of the current binary heap (i.e. the minimum value).

- The code sort.c is provided to you.

The data structures



Outline

- Priority Queue (ADT)
- A sorting based on heapsort (testing program given to you) Week 1

- Huffman coding (testing program written by you) Week 2

AN APPLICATION OF BINARY TREES - HUFFMAN CODING

Encoding

Application Layer Information


In July 1845 Texas formally accepted an American proposal to be annexed to the United States. Already between the United States and Mexico rapidly James K. Polk ordered General Zachary Taylo Corpus Christi. In March 1846, under instructi positions on the Rio Grande. On April 26 an of dragoons surrounded by Mexicans and unab out surrendered. Several Americans lost their liv

On May 8 the Mexicans intercepted Taylo were driven back. The next day Mexicans again at Resaca de la Palma. In June Taylor bega Monterrey, taking that city on September 25. the Americans took Saltillo and with little Tampico.

Santa Anna now took the field against the northern Mexico, finally engaging the American February. Upon learning of the Mexican's desig Wool, marched from San Antonio to join abandoning his prior plans to take Chihuahu bitter fighting Santa Anna pulled out his army control of northern Mexico.

While Taylor pursued the enemy, Colonel took the "Army of the West" into New Mexi Fe on August 16, 1846. Kearny then divided his to California and sending the remainder un Doniphan against Chihuahua. After General Winf Mexico City the two countries finally reached a

Source: Maps adapted from *The We* ican Wars, Volume 1, Freds

Text Video image Audio

Encoding

Low layer Bits

An application - Encoding

- An example – using only 1s and 0s to represent the following cases.
  - True or false
  - 4 directions: south, north, east and west

- ASCII is a fixed width encoding. Every character requires 8 bits to encode.

An application - Encoding

- Motivation: Can we save some bits in encoding?
- Thoughts:
  - some characters (like e and n) occur frequently, whereas characters (like q and z) occur very infrequently.
  - Do you still remember probability search (move to front heuristic)?

One Example

- Ex: “the fat cat sat on the mat”
  - If we use fixed-length binary code to represent the above sentence, what will that be? 26*8
  - Can we save some bits by using fewer bits for more frequent letters?
    - Frequency of letters:
      - a: 4, c:1 , e:2 , f:1 , h:2 , m: 1, n: 1, o: 1, s: 1, t: 6, “ ”:6
    - Bits of letters:
      - t & “ “ , frequently occur - 1 bit
      - a, h, e, less frequently – 2 bits
      - Etc.

Huffman Coding

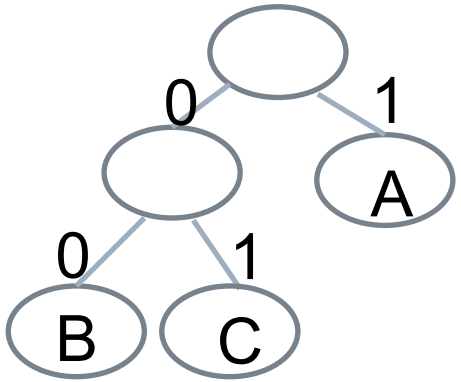
- Huffman Coding: variable width encoding
  - ❑ We are likely to save on the total number of bits required to encode a file/document if we represented frequently occurring character with fewer bits and infrequently occurring characters with more than 8 bits.

- Question: How do we know, given a variable length encoding, where characters stop and start?
  - ❑ ex: if e=11, z=1111 … so what's 111111?

- Answer: We use unique prefixes. Each character will have its own unique prefix. => How to implement?
  - ❑ We build a binary tree. By having only leaf nodes represent characters, we are able to guarantee unique prefixes.

Huffman Tree

- Use binary trees (i.e. Huffman tree). e.g. ‘ABCA’



- Q: How do we determine the optimal encoding given the frequencies of the characters?

- A: Use the leaf nodes that are farther away from the root to represent characters appear less frequently.

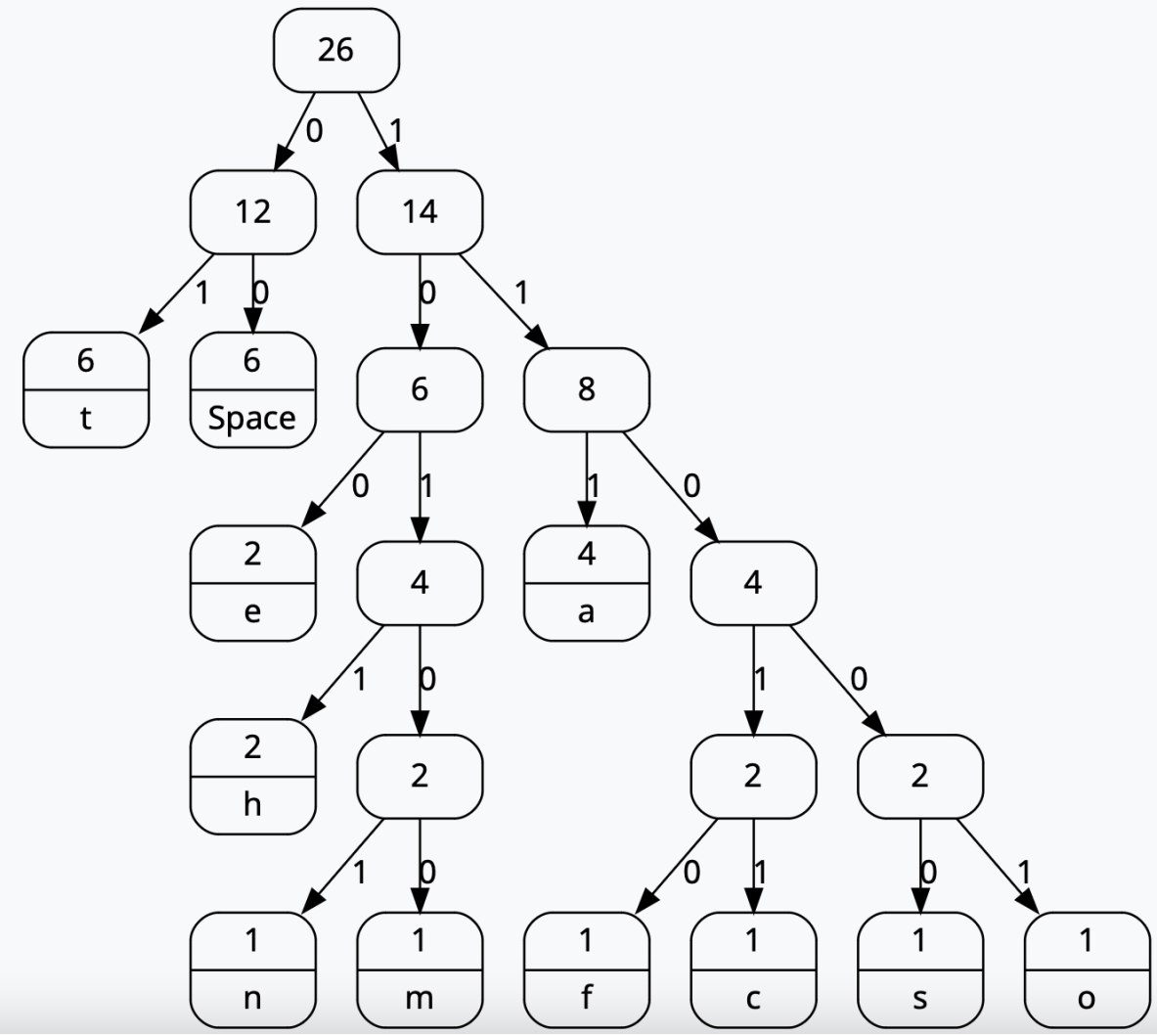
Huffman Tree

- <so now you have the “Huffman tree”>
- <label the left and right sub-branches of every node as 0 and 1 respectively>
- <it may be optimal, but is not necessarily unique>

Huffman Coding

- Ex: “the fat cat sat on the mat”
  - a: 4, c:1 , e:2 , f:1 , h:2 , m: 1, n: 1, o: 1, s: 1, t: 6, “_”:6
  - <create a node for each letter, and sort according to their values>
  - <remove two minimum nodes, combine with a new node, and insert this node with a new weight - the combined value of its two child nodes>
  - Is it better than the fixed length encoding?

Huffman Coding Cont'd



- Ex: "the fat cat sat on the mat"
  - a: 4, c:1 , e:2 , f:1 , h:2 , m: 1, n: 1, o: 1, s: 1, t: 6, "_":6
- Encoding:

Character	Encoding	Character	Encoding
a	111	n	10101
c	11011	o	11001
e	100	s	11000
f	11010	t	01
h	1011	"_"	00
m	10100		

- pack.h and pack.c are given

- Struct node {
- struct node* parent;
- int count;};

- Void pack(char* inflie, char* outfile, struct node* nodes[257]);