CSC263

Binary Heap

Binomial Heap

AVL Tree

Hash - Bloom Filter - Quick Sort

Amortized

Disjoint Set

Graph

BFS

DFS

MST

Decision Tree

Binary Heap:

Binary Heap 题型

1. 2015 may Question1

Question 1. [10 MARKS]

Consider a **max**-heap of four items with distinct keys k_1, k_2, k_3, k_4 where $k_1 < k_2 < k_3 < k_4$.

Part (a) [4 MARKS]

Draw all the different max-heaps that are possible for the elements described above.

Part (b) [6 MARKS]

Suppose that all the arrangements from the previous part are equally likely, and that we insert a random item whose key is equally likely to be in any order with respect to k_1, k_2, k_3, k_4 (in between any two of them or beyond either endpoint).

Calculate the average **number of swaps** that would take place for this one INSERT operation.

2. Assignment 2 Question 6

The task in this question is to compute the medians of all prefixes of an array. As input we are given the array A[1 . . n] of arbitrary integers. Using a heap data structure, design an algorithm that outputs another array M[1 . . n], so that M[i] is equal to the median of the numbers in the subarray A[1 . . i]. Recall that when i is odd, the median of A[1 . . i] is the element of rank (i + 1)/2 in the subarray, and when i is even, the median is the average of the elements with ranks i/2 and i/2 + 1. Your algorithm should run in worst-case time O(n log n). a. Describe your algorithm in clear and concise English, and also provide the corresponding pseudocode. Argue that your algorithm is correct. b. Justify why your algorithm runs in time O(n log n).

Amortized

Amortized 题型

1.2015 dec Question4

4. (10 marks) Suppose we implement min-heap as an array. To perform an INSERT operation, if the array is not full we proceed as explained in class. If the array is full, we first move the elements of the heap from the current (full) array into a new array of twice the size and then we perform the insertion in the new (half-full) array. To perform an EXTRACT-MIN operation, if the array is more than a quarter full we proceed as explained in class. If the array is only a quarter full, we first move the elements of the heap from the current (quarter-full) array into a new array of half the size, and then we perform the EXTRACT-MIN in the new (half-full) array. Suppose that the heap is initially empty and we perform an arbitrary sequence of n INSERT and EXTRACT-MIN operations.

(a) What is the worst-case cost for an individual operation in such a sequence? Explain your answer.

(b) Use the accounting method to determine the amortized cost per operation in a sequence of n operations. State the cost of each operation, and the credit invariant, and explain why the credit invariant can be maintained.

2. Assignment 4

Question 2. (20 marks) A manufacturing company producing Product X maintains a display showing how many Product X's they have manufactured so far. Each time a new Product X is manufactured, this display is updated. Interestingly, this company prefers base 3 notation; instead of using bits (0 and 1) as in base 2 notation, they use trits (0, 1 and 2) to represent this number. The total cost of updating the display is $(c + dm), where c is a fixed cost to open up the display, d is the cost of changing each display digit, and m is the number of digits that must be changed. For example, when the display is changed from 12201222 to 12202000 (because 12201222 + 1 = 12202000 in base 3) the actual cost to the company is $(c + 4d) because 4 digits must be changed. The manufacturing company wants to amortize the cost of maintaining the display over all of the Product X's that are manufactured, charging the same amount to each. In other words, we are interested in the amortized cost of incrementing the display by one. Let A(n) be the amortized cost per display change operation when we perform a sequence of n successive display changes, starting from the number 0. For example, A(3) = c + 4 / 3 d. In this question, we want you to give an upper bound on A(n).

More precisely, consider the list: c + 4/ 3 d, c + 17 /12 d, c + 3 / 2 d, c + 7/ 4 d, c + 2d, c + 5 / 2 d (note that the numbers in this list are sorted from smallest to largest). What is the smallest cost B in the above list such that $A(n) \leq B$ for all $n \geq 1$? Prove the correctness of your answer in two ways: first analyses A(n) using the aggregate method, and then analyses A(n) using the accounting method with an appropriate charging scheme. Make sure you justify why your answer is the smallest such B from the above list.

AVL Tree

AVL Tree 题型

1. 2015 Dec Question 1

1. (14 points, 2 points each) Circle the correct answer; no justification is necessary.
(a) True or False: Let X and Y be two random variables. Then E[X + Y] = E[X) + E[Y).

(b) True of False: The worst case complexity is O(logn) for performing a search in an AVL tree.

(c) True or False: The worst case complexity is O(logn) for search in a binary search tree.

(d) True or False: The operations of delete and insert applied to a binary search tree are commutative. That is, given a binary search tree T, deleting x and then delete y leaves the same tree as deleting y and then x.

(e) Suppose we have n keys and they are searched for with equal probability. They are stored in a binary search tree that minimizes the expected number of nodes examined to find a key. The worst-case number of nodes examined to find a key in that tree is:
i. ©(1)
ii. ©(n)
iii. ©(log (ת
iv. None of the above.

(f) Suppose we insert n keys into a binary search tree. The first inserted key is 0, the second one is 1, and from then on, each inserted key is the average of the previous two. Thus the sequence of inserted keys begins as follows: 0 , 1 , ^ , | , | , The height of the resulting tree is:
i. 0(1)
ii. ©(logn)
iii. 0(n)
iv. ©(nlogn)
v. None of the above.

(g) Let G be a connected, undirected graph with n nodes and m edges. Suppose its edges have distinct weights 1,2,..., m. Then the cost of a minimum cost spanning tree of G must be exactly:
i. n — 1
ii. n
iii. n(n — l)/2
iv. n (n + l)/2
v. None of the above.

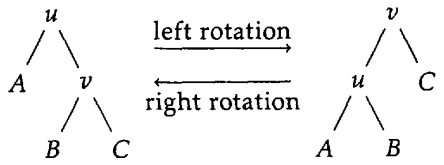
2014 Dec Question 5

Question 5. [7 MARKS]

Suppose an AVL tree is augmented so that each node x contains a pointer $Lr[x]$ to the leftmost node in its right subtree. If the right subtree of x is empty, then $Lr[x] = \text{NIL}$.

Part (a) [1 MARK]

Explain how to update $Lr[v]$ in constant time when a left rotation is performed on u and its right child v or when a right rotation is performed on v and its left child u, as illustrated on the right.



Part (b) [2 MARKS]

Explain how to update $Lr[u]$ in constant time when a left rotation is performed on u and its right child v or when a right rotation is performed on v and its left child u, as illustrated above.

Question 5. (CONTINUED)

Part (c) [4 MARKS]

Explain how to maintain Lr in every node during an INSERT(x) operation, without affecting the $\mathcal{O}(\log n)$ worst-case runtime of INSERT.

Hash + Bloom Filter + Quick Sort

Hash 题型
2015 may Question 6

Question 6. [15 MARKS]

Consider the following INVENTORY ADT used to model the products sold on a shopping website.

Objects: Sets of *products*. Each product P consists of the following attributes: $P.code \in \mathbb{Z}^+$ (a **unique** positive integer *product code*) and $P.rating \in [0, 10]$ (a **real number** *rating* in the interval $[0, 10]$).

Operations:

- GETPRODUCT(S, C): return a product $P \in S$ such that $P.code = C$; return NIL if there is **no** product in S with product code C.
  Requirement: runs in average-case time $\mathcal{O}(1)$.
- ADDPRODUCT(S, P): add product P to set S; precondition: S contains **no** other product with product code $P.code$—*do NOT check this condition, just assume it.*
  Requirement: runs in worst-case time $\mathcal{O}(\log n)$.
- TOPRATED(S, r): return the **number** of products in set S with a rating $\geqslant r$.
  Requirement: runs in worst-case time $\mathcal{O}(\log n)$—*part marks for* $\mathcal{O}(n)$.

Part (a) [1 MARK]

What data structure would you use to achieve $\mathcal{O}(1)$ average-case runtime for operation GETPRODUCT?

Part (b) [1 MARK]

Write an implementation for operation GETPRODUCT. State your assumptions clearly.

Part (c) [4 MARKS]

What data structure would you use to achieve $\mathcal{O}(\log n)$ worst-case runtime for operation TOPRATED? Describe clearly what information is stored in each element of your data structure.

Question 6. (CONTINUED)

Part (d) [4 MARKS]

Write an implementation for operation AddProduct (in clear English). State your assumptions clearly.

Part (e) [5 MARKS]

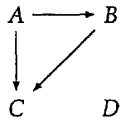
Write a *detailed* implementation for operation TopRated, **in pseudo-code**. State your assumptions clearly.

Graph

Graph 题型
1. 2015 may Question 3

Question 3. [15 MARKS]

The same graph can be represented by a number of different data structures. Consider the directed graph G pictured on the right. The same directed graph is represented in four different ways below (some of which we have studied and some others we have not).



Adjacency Matrix: Vertices $[A, B, C, D]$.

0	1	1	0
0	0	1	0
0	0	0	0
0	0	0	0

Regular Adjacency Lists: (Every vertex u has a list of the vertices v such that G contains an edge (u, v) *from u to v.*)

- $A : [B, C]$
- $B : [C]$
- $C : []$
- $D : []$

***Inverted* Adjacency Lists:** (Every vertex u has a list of the vertices v such that G contains an edge (v, u) *from v to u.*)

- $A : []$
- $B : [A]$
- $C : [A, B]$
- $D : []$

List of vertices and unsorted list of edges:

$V = [A, B, C, D]$
$E = [(A, B), (A, C), (B, C)]$

Below is a table listing five operations that could be performed on a graph. For each of the representations above, give the asymptotic worst-case complexity of the operation expressed in terms of $|V|$ and $|E|$. The first operation has been completed for you as an example.

For every representation, assume that it is possible to find the *size* of any list (or matrix) in $\mathcal{O}(1)$ time. You only need to put your final answer in the table—no justification is required.

Operation	Adjacency Matrix	Regular Adjacency Lists	Inverted Adjacency Lists	List of Vertices and Edges
edge query	$\mathcal{O}(1)$	$\mathcal{O}(\min(\|V\|, \|E\|))$	$\mathcal{O}(\min(\|V\|, \|E\|))$	$\mathcal{O}(\|E\|)$
Determine if a path of length 2 exists from u to v				
Return a list of in-neighbours of vertex v				
Return a list of the self-loops				
Return the number of edges				

BFS

BFS 题型
2015 May Question 8

Question 8. [15 MARKS]

A *word ladder* is a sequence of English words, where two adjacent words differ by only one letter. For example:

COLD → CORD → CARD → WARD → WARM

You are given the two words "PHYSICS" and "GEOLOGY", and you must design an algorithm that computes a word ladder between the two words, with the **shortest possible** sequence of words.

Input: The two words "PHYSICS" and "GEOLOGY", and an **unsorted** list of 7-letter English words. Assume that there are n such words in the list, and that all words consist of **only** the 26 English alphabet letters.

Output: A word ladder sequence from "PHYSICS" to "GEOLOGY" that contains the smallest possible number of words. If such a word ladder is impossible, return NIL.

Requirement: The average-case runtime of the algorithm must be in $\mathcal{O}(n)$.

Part (a) [4 MARKS]

How do you model this problem using a graph? Describe what each vertex represents and what each edge represents. Indicate whether it is a directed or undirected graph.

Part (b) [2 MARKS]

In order to be able to construct your graph from Part (a) in $\mathcal{O}(n)$ average-case runtime, you need to use a data structure to store the list of 7-letter English words. What data structure should you use?

Question 8. (CONTINUED)

Part (c) [4 MARKS]

Give a clear description of the construction process of your graph from Part (a) and justify that it takes average-case runtime $\mathcal{O}(n)$.

Part (d) [5 MARKS]

Using the graph you constructed in Part (c), describe in clear English how to find a word ladder with the shortest sequence of words. Justify briefly that your algorithm is correct and has the desired runtime.

DFS

DFS 题型

2015 dec Question 8

8. (12 marks) Recall that a simple cycle in an undirected graph G = (V, E) is a sequence of distinct vertices v0, v1, v2, …., vk-1 where k > 3 and (üo, u!} G E, {v0 , v1} ∈ E, … , (vk-1, v0} ∈ E.

An undirected graph G is acyclic (i.e., it contains no simple cycle) if and only if G is a forest (i.e., G consists of one or more tree(s)).

Explain how to use a graph algorithm discussed in class to determine, in 0(|V|) worst case time, whether a given undirected graph G = (V, E) contains a cycle. State which algorithm from class you will use, and then how to use it to solve the cycle problem. Briefly explain why your solution runs in 0(|V|) time.

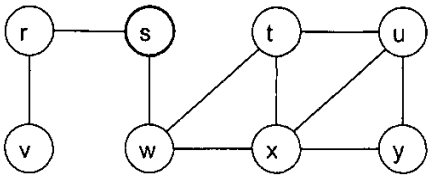
MST

MST 题型

2015 dec Question 5

5. BFS/DFS problem (12 points)

Assume that the graph above is given as an array of adjacency lists where the vertices are listed in alphabetical order, and for each vertex, the vertices in its adjacency list are listed in alphabetical order. For example, the adjacency list for vertex t contains the vertices u, w, x, in this order.



(a) Execute the BFS algorithm on the following undirected graph starting from vertex s. Draw the resulting BFS tree. What is the distance label d[u] of vertex u in this BFS?

(b) Now execute the DFS algorithm on the same graph starting from vertex s. Draw the resulting DFS forest. What are the discovery and finish times of vertex u in this DFS? Draw the back, forward and cross edges (if any) with dotted lines and label them as back, forward, cross accordingly.

Disjoint Set

Disjoint Set 题型
2015 dec Question 7

7. (16 points) Consider the following ADT that consists of a subset $S \subseteq \{1, \ldots, n\}$ and supports the following operations:

   - DELETE(S, i): Delete integer $i \in \{1, \ldots, n\}$ from the subset S. If $i \notin S$, this operation has no effect.
   - PRED(S, i): Return the predecessor in S of $i \in \{1, \ldots, n\}$, i.e. $\max\{j \in S \mid j < i\}$. If i has no predecessor in S (i.e. if $i \leq \min(S)$), then return 0.

   Initially, $S = \{1, \ldots, n\}$.

   In this question, you will explain how to use a disjoint set data structure to implement this ADT so that the amortized cost of each operation is in $O(\log^* n)$.

   (a) (1 point) Let $n = 12$ and $S = \{3, 5, 6, 8, 12\} \subseteq \{1, \ldots, 12\}$.

   What is the set $\{i \in \{1, \ldots, 12\} \mid \text{PRED}(S, i) = 0\}$?

   What is the set $\{i \in \{1, \ldots, 12\} \mid \text{PRED}(S, i) = 5\}$?

   What is the set $\{i \in \{1, \ldots, 12\} \mid \text{PRED}(S, i) = 8\}$?

   What is the set $\{i \in \{1, \ldots, 12\} \mid \text{PRED}(S, i) = 12\}$?

   (b) (4 points) Explain how the disjoint set abstract data type can be used to represent a subset $S \subseteq \{1, \ldots, n\}$. Include a diagram of your data structure for the subset $S = \{3, 5, 6, 8, 12\} \subseteq \{1, \ldots, 12\}$. (Hint: Look at part (a).)

   (c) (1 point) Which disjoint data set structure should you use?

(d) (3 points) Explain how to implement DELETE(S, i).

(e) (3 points) Explain how to implement PRED(S, i).

(f) (2 points) Briefly explain why the amortized cost of each operation is in $O(\log^* n)$.

(g) (2 points) Explain how to initialize your data structure. How much time does it take?