**Examination 3**

**CSC301 Algorithms and Data Structures**

**19 April 2018**

**1. FORTRAN was the first high-level language. FORTRAN is as old as**

**which object in our classroom?**

**2. In what way was the mention of FORTRAN relevant to the dynamic**

**connectivity problem?**

There are similarities in the dynamic connectivity problem.

**3. Which three qualities must a relation possess to be an equivalence relation?**

It must be reflexive, symmetric, and transitive.

**4. In a word or two, describe these Linux commands:**

* 1. cd -change directory
  2. Cp – copy
  3. Ls – list available files and directoires
  4. Mv – move
  5. Pwd – print working directory
  6. Rm – remove

**5. My favorite definition of computer science includes four questions. To**

**which of the four questions does each of the following words correspond?**

1. **•**Computability – quick union
2. Complexity – quick find
3. Compatability – weighted quick union

**6. What do the following Linux commands accomplish?**

**j a v a c u f /UF. j a v a**

**j a v a u f .UF < tinyUF . t x t**

**7. Describe the format of the data files that we are using in our union-find**

**experiments.**

**8. The authors of our textbook developed a sequence of progressively better**

**solutions to the dynamic connectivity problem.**

**(a) What is the complexity of the first (least efficient) solution?**

**1**

**(b) What is the complexity of the last (most efficient) solution?**

**9. How might you use these features of NetBeans?**

**(a) source/format**

**(b) source/fix imports**

**(c) refactor/rename**

**(d) refactor/encapsulate fields**

**10. What is an abstract class?**

An abstract class can’t be instantiated, but can be subclasses.

**1. In just a few words, what is an important way in which the tilde analysis**

**differs from a Big-Oh analysis?**

Tilde analysis takes into account the lower bound of runtime and is more of an average than a worst-case scenario.

**2. What is a “wide API?” Is a wide API a good thing? Why or why not?**

**3. What is an “amortized analysis?” For which of the data structures that**

**we have examined is amortized analysis appropriate?**

**4. If we want to define a class that models a Bag using a linked list, then we**

**might find it convenient to use inner classes for which two purposes?**

We would use inner classes for the ability to add to or to remove from the list primarly.

**5. Describe the API for a data structure that supports the FIFO protocol.**

**FIFO is the format of the stack data structure. It has a pop and push functions that allow the client to add an element to the top of the stack or take an element from the top of the stack.**

This structure would be a stack data structure. It puts items in and then removes the most recently item that was added.

**6. Write a method that finds the value of the smallest number in an array of**

floating point numbers.

Public static int(array input){

Int result = input[0];

For(int i: input){

If(input[i] > result){

Result = input[i];

}

Return result;

}

**7. Write a method that finds the position (index) of the smallest number in**

**an array of floating point numbers.**

**8. Write a method that finds the position (index) of the smallest number**

**in that part of an array of floating point numbers that begins with the**

**element at a specified position.**

**9. Write a method that copies the value it finds at position i in an array of**

**floating point numbers to position j, while it also copies the value found**

**at position j to position i.**

**10. Write a method whose parameters are an array of floating point numbers**

**and two indices i and j of elements in that array. This method will. . .**

**• store the value found at position i in variable temp**

**• move all of the values in positions (i − 1) down to j one place to the**

**right**

**• write the value in temp into position j**

**11. Write a method whose parameters are an array of floating point numbers**

**and the index i of an element of that array. This method will. . .**

**2**

**Figure 1: UML class diagram.**

**• Examine elements of the array beginning with the element at position**

**j = (i − 1) and continuing leftward to j = (i − 2), j = (i − 3), and**

**so on until it either reaches the element at position j = 0 or an**

**element whose left neighbor has a value less than that of the element**

**at position i.**

**• Return j.**

**12. The choice to implement an interface obliges a programmer to define the**

**methods of the interface within the class. Which methods must the program**

**define in these cases?**

**(a) public class Bag<ElementType> implements Iterable<ElementType> { }**

**(b) public class BagIterator<ElementType> implements Iterator<ElementType> { }**

**13. The complexity of the shell sort lies somewhere between N log N and N2**

**.**

**Give one of the expressions that the authors of our textbook gave us to**

**describe the complexity of the shell sort.**

**14. This list is H-sorted: [1 2 3 3 6 5 10 7 15 11 21 13 28 17]. In this**

**case, H = 2. What does H-sorted mean?**

H-sorted means that values that are within a range of two from each other may not be sorted.

**15. Write a method whose one parameter is an integer n and whose return**

**type is List<Double>. This method will return to its caller a list of n**

**random floating point values that are sorted in ascending order.**

**Suggestion: Find a method in the java. util .Random class that will generate**

**random floating point values. Find a method in the java. util . Collections**

**class that will sort the elements of a list.**

**16. Write a method whose two parameters are both sorted lists of floating**

**point values. This method will merge the two lists—it will return to its**

**caller a sorted list of floating point values that contains all the values that**

**it finds in the two given lists.**

**Suggestion: Declare three integer variables in the method. Each integer**

**will serve as an index into one of the three lists (the two parameters and**

**3**

**the list that is returned) on which the method works. Use three while**

**loops. One will do most of the work. The other two are needed to “clean**

**up” at the end.**

**17. Complete the following table. Approximate values will suffice. (Approximate**

**by truncating.)**

**N log2 N N2 N log2 N**

**1 0 1 0**

**10 ≈ 3 102 ≈ 3 · 10**

**102 ≈ 6 104 ≈ 6 · 102**

**103**

**106**

**109**

**1012**

**N log2 N N2 N log2 N**

**1 0 1 0**

**10 ≈ 3 102 ≈ 3 · 10**

**102 ≈ 6 104 ≈ 6 · 102**

**103 ≈ 10 106 ≈ 10 · 103**

**106 ≈ 20 1012 ≈ 20 · 106**

**109 ≈ 30 1018 ≈ 30 · 109**

**1012 ≈ 40 1024 ≈ 40 · 1012**

**18. There are places in our textbook where the authors describe opportunities**

**to reduce the time required for a computation by 30%. However, in other**

**places they emphasize that our ability to develop, analyze, and select**

**algorithms can have a much bigger impact. Explain.**

**Sometimes choosing an algorithm poorly will make it impossible to solve**

**the problem. Finding the right algorithm might be the prerequisite to**

**putting a solution in reach.**

**19. Within the selection sort algorithm there is an algorithm that searches.**

**There is also a search inside of the insertion sort. How do the two searches**

**differ?**

**The search in the selection sort proceeds from left to right through the**

**still unordered part of the list. It must examine every element in the unordered**

**part to the list in order to identify the element with the smallest**

**value.**

**4**

**The search in the insertion sort proceeds from right to left through the**

**already sorted part of the list. It does not need to examine every element**

**in this part of the list. It quits as soon as it finds the right place in which**

**to insert the next element.**

**20. What are the three defining properties of an equivalence relation?**

**An equivalence relation is reflexive, symmetric, and transitive.**

**For all elements a, b, and c:**

**• a is related to a**

**• If a is related to b, then b is related to a.**

**• If a is related to b and b is related to c, then a is related to c.**

**21. What is the significance of <ElementType> in this code?**

**A programmer will substitute the name of some specific class (a specific**

**type) for <ElementType> when creating an instance of the Bag class. In**

**this way, the programmer specifies what kinds of objects the Bag may**

**hold.**

**package bag ;**

**import j a v a . u t i l . I t e r a t o r ;**

**public c l a s s Bag<ElementType> implements I t e r a b l e <ElementType> {**

**private Node<ElementType> anchor ;**

**private int s i z e ;**

**public Bag ( ) {**

**th is . anchor = nu l l ;**

**th is . s i z e = 0 ;**

**} // Bag ( )**

**public boolean isEmpty ( ) {**

**return th is . s i z e == 0 ;**

**} // isEmpty ( )**

**public int s i z e ( ) {**

**return th is . s i z e ( ) ;**

**} // s i z e ( )**

**5**

**public void add ( ElementType v al u e ) {**

**th is . anchor = new Node ( v alue , th is . anchor ) ;**

**th is . s i z e ++;**

**} // add ( ElementType )**

**public Node getAnchor ( ) {**

**return th is . anchor ;**

**} // ge tAnchor ( )**

**@Override**

**public I t e r a t o r <ElementType> i t e r a t o r ( ) {**

**return new B a g I t e r a t o r ( th is ) ;**

**} // i t e r a t o r ( )**

**} // Bag ( )**

**22. I did not have to type every line of this code. What part was NetBeans**

**able to fill in for me?**

**NetBeans wrote the package statement and the getters (the accessors—**

**getValue() and getNext()). It also anticipated my needs for closing braces**

**and parentheses.**

**package bag ;**

**public c l a s s Node<ElementType> {**

**private f i n a l ElementType v al u e ;**

**private f i n a l Node nex t ;**

**public Node ( ElementType v al u e ) {**

**th is . v al u e = v al u e ;**

**th is . nex t = nu l l ;**

**} // Node ( ElementType )**

**public Node ( ElementType v alue , Node nex t ) {**

**th is . v al u e = v al u e ;**

**th is . nex t = next ;**

**} // Node ( ElementType , Node )**

**public ElementType ge tV alue ( ) {**

**return th is . v al u e ;**

**} // g e tV al u e ( )**

**6**

**public Node getNext ( ) {**

**return th is . nex t ;**

**} // g e tN e x t ( )**

**} // Node**

**23. Complete the definition of the BagIterator class.**

**package bag ;**

**import j a v a . u t i l . I t e r a t o r ;**

**public c l a s s B a g I t e r a t o r<ElementType> implements I t e r a t o r <ElementType> {**

**private f i n a l Bag bag ;**

**private Node<ElementType> cu r ren tNode ;**

**public B a g I t e r a t o r ( Bag bag ) {**

**th is . bag = bag ;**

**th is . cu r ren tNode = th is . bag . getAnchor ( ) ;**

**} // B a g I t e r a t o r ( Bag )**

**@Override**

**public boolean hasNext ( ) {**

**// Th is i s a s t u b method−−−c om ple te i t s d e f i n i t i o n .**

**return f a l s e ;**

**} // hasNex t ( )**

**@Override**

**public ElementType next ( ) {**

**// Th is i s a s t u b method−−−c om ple te i t s d e f i n i t i o n .**

**return nu l l ;**

**} // n e x t ( )**

**} // B a g I t e r a t o r**

**@Override**

**public boolean hasNext ( ) {**

**return th is . cu r ren tNode != nu l l ;**

**} // hasNex t ( )**

**@Override**

**public ElementType next ( ) {**

**ElementType v al u e = th is . cu r ren tNode . ge tV alue ( ) ;**

**th is . cu r ren tNode = th is . cu r ren tNode . getNext ( ) ;**

**return v al u e ;**

**7**

**} // n e x t ( )**

**24. Here is code that creates a Bag and adds several integers to it. Add code**

**that will print the values of the integers in the Bag.**

**(The Bag class implements the Iterator interface.)**

**Bag<I n t e g e r > moonLandings = new Bag<>():**

**moonLandings . add ( 11 ) ;**

**moonLandings . add ( 12 ) ;**

**moonLandings . add ( 14 ) ;**

**moonLandings . add ( 15 ) ;**

**moonLandings . add ( 16 ) ;**

**moonLandings . add ( 17 ) ;**

**for ( int n : moonLandings ) {**

**System . out . p r i n t l n ( n ) ;**

**} // f o r**

**// or , we c o ul d do i t t h i s way . . .**

**I t e r a t o r <I n t e g e r > i t e r a t o r = moonLandings . i t e r a t o r ( ) ;**

**while ( i t e r a t o r . hasNext ( ) ) {**

**int n = i t e r a t o r . nex t ( ) ;**

**System . out . p r i n t l n ( n ) ;**

**} // w h i l e**

**25. Complete the code for the selection sort.**

**public s t a t i c int positionO fMinimum ( int [ ] data , int i ) {**

**int bes tGuessSoFar = i ;**

**for ( int j = i ; j < data . l e n g t h ; j++) {**

**i f ( data [ j ] < data [ bestGuessSoFar ] ) {**

**bes tGuessSoFar = j ;**

**} // i f**

**} // f o r**

**return bes tGuessSoFar ;**

**} // pos it ionO fM in imum ( i n t [ ] , i n t )**

**public s t a t i c void swap ( int [ ] data , int i , int j ) {**

**int temp = data [ i ] ;**

**data [ i ] = data [ j ] ;**

**data [ j ] = temp ;**

**} // swap ( i n t [ ] , i n t , i n t )**

**8**

**public s t a t i c void s e l e c t i o n S o r t ( int [ ] data ) {**

**for ( int i = 0 ; i < data . l e n g t h ; i++) {**

**// Code i s needed he re .**

**} // f o r**

**} // s e l e c t i o n S o r t ( i n t [ ] )**

**} // InOrder**

**public s t a t i c void s e l e c t i o n S o r t ( int [ ] data ) {**

**for ( int i = 0 ; i < data . l e n g t h ; i++) {**

**int j = positionO fMinimum ( data , i ) ;**

**swap ( data , i , j ) ;**

**} // f o r**

**} // s e l e c t i o n S o r t ( i n t [ ] )**

**26. Complete the code for the insertion sort.**

**public s t a t i c int i n s e r t i o n P o i n t ( int [ ] data , int i ) {**

**int j = i ;**

**while ( j > 0 && data [ j − 1 ] > data [ i ] ) {**

**j −−;**

**} // w h i l e**

**return j ;**

**} // i n s e r t i o n P o i n t ( i n t [ ] , i n t )**

**public s t a t i c void i n s e r t ( int [ ] data , int i , int j ) {**

**int temp = data [ i ] ;**

**for ( int k = i ; k > j ; k−−) {**

**data [ k ] = data [ k − 1 ] ;**

**} // f o r**

**data [ j ] = temp ;**

**} // i n s e r t ( i n t [ ] , i n t , i n t )**

**public s t a t i c void i n s e r t i o n S o r t ( int [ ] data ) {**

**for ( int i = 0 ; i < data . l e n g t h ; i++) {**

**// Code i s needed he re .**

**} // f o r**

**} // i n s e r t i o n S o r t ( i n t [ ] )**

**9**

**public s t a t i c void i n s e r t i o n S o r t ( int [ ] data ) {**

**for ( int i = 0 ; i < data . l e n g t h ; i++) {**

**int j = i n s e r t i o n P o i n t ( data , i ) ;**

**i n s e r t ( data , i , j ) ;**

**} // f o r**

**} // i n s e r t i o n S o r t ( i n t [ ] )**

**27. A heap is a kind of a tree. We can use an array to hold the elements of**

**the heap and indices on that array to locate the nodes of the tree.**

**(a) If k is the index a node, what is the index of that node’s left child?**

**(b) If k is the index of a node, what is the index of that node’s right**

**child?**

**(c) If k is the index of a node, what is the index of that node’s parent?**

**(d) What is the ordering rule that relates the value in a node to the**

**values in its children?**

**28. We can use one data structure to make another. For example, we built**

**stacks and queues on top of arrays and linked lists.**

**(a) Explain how a priority queue could be used as the basis for a stack.**

**(b) Explain how a priority queue could be used as the basis for a queue.**

**29. Draw the binary search tree that results when integers are added to an**

**empty tree in the following order: 4–2–1–3–6–5–7**

**30. Draw the binary search tree that results when integers are added to an**

**empty tree in the following order: 1–2–3–4–5–6–7**

**31. What is the minimum height of a binary search tree that contains N**

**elements?**

A minimum height of 2

**32. What is the maximum height of a binary search tree that contains N**

**elements?**

A maximum height of N

**33. A binary search within an ordered array requires logarithmic time. A**

**search in a binary search tree that has been constructed by adding random**

**values also requires logarithmic time.**

**(a) Which kind of search is faster?**

**(b) Insertions are faster on which kind of data structure?**

**34. What does the floor () method for the binary search tree (shown on page**

**407) accomplish?**

**35. How are the rank() and select () methods for the binary search tree (shown**

**on page 409) related?**

**10**

**36. What do the authors of our textbook have to say a about the relative**

**merits of using loops versus recursion in our definitions of the methods of**

**the binary search tree class?**

**37. There are two easy and one hard case for deletion of a node from a binary**

**search tree. What is the hard case?**

**38. The authors of our textbook introduce us to binary search trees, 2–3 trees,**

**and red-black trees, in that order.**

**(a) What advantage do 2–3 trees offer over binary search trees?**

**(b) How are red-black trees related to 2–3 trees? to binary search trees?**

**39. In a 2–3 tree, a 3-node with keys a and b has pointers to subtrees that**

**contain. . .**

**• elements whose keys are less than a**

**• elements whose keys have values between those of a and b**

**• elements whose keys are greater than b**

**In a red-black tree, two 2-nodes can serve the same purpose.**

**• The 2-node whose key is b has pointers to. . .**

**– a right subtree whose elements have what relation to b?**

**– a left subtree whose root node contains a and that has pointers**

**to. . .**

**∗ a left subtree whose elements have what relation to a?**

**∗ a right subtree whose elements have what relation to a and**

**b?**

**40. There are three constraints that define a red-black tree.**

**(a) All red links lean in which direction?**

**(b) No node has two links of which color connected to it?**

**(c) The tree has perfect balance when balance is measured in which way?**

**41. Because red-black trees are both binary search trees and 2–3 trees, they**

**have properties of both kinds of trees.**

**(a) Which valuable trait do the red-black trees inherit from binary search**

**trees?**

**(b) Which valuable trait to the red-black trees inherit from 2–3 trees?**

**42. Compare the code on page 399 of our textbook with the code on page 439.**

**What is the difference?**

**43. What are the three characteristics of a good hash functions?**

**11**

**44. There are two ways of resolving collisions in a hash table. Linear probing is**

**a form of open addressing. Separate chaining is the other method. Which**

**one of these two approaches makes use of linked lists?**

**45. Some operations on symbol tables can be efficient if the table is built on a**

**tree but not if it is build on a hash table. If an application requires these**

**operations, then a symbol table built on a hash table is not a good choice.**

**What are the operations that make a hash table a poor choice?**

**46. The textbook lists some important applications of symbol tables.**

**(a) What is a concordance?**

**(b) What is DNS (the Domain Name Service)?**

**(c) What is a .csv (comma separated value) file?**

**(d) What is a sparse vector?**

**47. We have a special word for a sequence of vertices that are connected by**

**edges. What is it?**

**48. We have a special word for a sequence of vertices that are connected by**

**edges in which the first and last vertex in the sequence are the same. What**

**is it?**

**49. We have a special word to name an acylic connected graph. What is it?**

**50. We have a special word to name a disjoint set of acyclic connected graphs.**

**What is it?**

**51. In a complete graph, there is an edge to connect every pair of vertices. If**

**the complete graph has V vertices, how many edges does it have?**

**52. If a graph G has V vertices and E edges, how many edges are in the**

**graph’s minimum spanning tree?**

**53. Here are the first few steps of Kruskal’s algorithm for the construction of**

**a minimum spanning tree for a graph G.**

**• Create an empty tree T.**

**• Select the edge e from G that has the least weight. (The weight**

**might be, for example, the length of the edge.)**

**• Add e to T if it does not create a cycle. (That is, add e if it is not**

**the case that the two vertices that are the edges endpoints are not**

**already part of T).**

**(a) What action(s) must be repeated?**

**(b) These action(s) must be repeated until what condition is satisfied?**

**54. Here are the first few steps of Prim’s algorithm for the construction of a**

**minimum spanning tree for a graph G.**

**12**

**• Create an empty tree T.**

**• Create an empty collection C of edges.**

**• Select an vertex v from G.**

**• Add all of the edges that connect to v to C.**

**• Select the edge e in C that has the smallest weight.**

**• Add e to T.**

**• The edge e connects the vertex v to some other vertex u. Add to C**

**all of the edges that connect to u.**

**(a) What action(s) must be repeated?**

**(b) These action(s) must be repeated until what condition is satisfied?**

**55. Dijkstra’s shortest path algorithm finds which two pieces of information**

**for each vertex in a graph?**

**56. A Java programmer who chooses to override the definition of the hashcode()**

**method in a class must also override the definition of which other method?**

**57. What results from the addition of an edge to a tree? (Here the word**

**“addition” means the creation of a new edge to connect vertices that are**

**already in the tree—it does not increase the number of vertices in the**

**tree.)**

**58. What results from the deletion of an edge of a tree? (Here the word**

**“deletion” means the removal of a connection between two vertices—the**

**vertices remain a part of the tree.)**

**59. What is an “amortized analysis” of the complexity of an algorithm?**

**60. Why might we expect to find a logarithmic term in the characterization**

**of the complexity of a divide-and-conquer algorithm?**

**61. What is the signficance of the inner loop to us when we set out to measure**

**the complexity of an algorithm?**

Inner loops tend to add significantly to the complexity of an algorithm. It usually results in squaring of the difficulty of the algorithm.

**62. You will work on teams. You will have to collaborate. You will have to**

**present your work to others. You will pass your work on to colleagues who**

**will develop it further. Testing and documentation will be a part of your**

**work from the start of a project to the finish.**

**In a paragraph or two, explain how you might use these tools.**

**• assert statements**

**• Graphviz**

**• HTML (especially <a> and <img> tags)**

**• Javadoc**

**13**

**• JUnit**

**• LATEX(especially the beamer class)**

**• Logger class in the Java API**

**• Mercurial**

**• UML class diagrams**

I would use these tools to keep track of everything that I have changed in a project. This is will allow others to be able to revert anything that I have done in the case that something becomes broken from my contribution. It will also give me the opportunity to explain the logic used in the changes that I have made in order to ensure there is a consensus on the changes.

**63. Thirty-five years ago last Sunday, a tank on the Apollo 13 spacecraft**

**exploded. The oxygen that was to have sustained the astronauts during**

**their round-trip journey between earth and moon was lost. If the tank**

**had not exploded, the astronauts would have had the oxygen to breath,**

**to generate electrical power, and to make water for cooling and for their**

**own consumption.**

**I showed you two excerpts from a motion picture that told the story.**

**In the first excerpt, we saw the flight director addressing his team. They**

**faced many challenges. Each member of team had different responsibilities,**

**different concerns, and different ideas. A young engineer stepped**

**forward. He confidently explained that they must immediately shut down**

**almost everything on the ship. He might have been the youngest person**

**in the room, but he knew his job very well. The flight director and the**

**other engineers listened.**

**Engineering is a team effort. Every member of a team is important.**

**In the second excerpt, the astronauts needed a filter to remove poisonous**

**carbon dioxide from the air on their ship. Engineers had to design a filter**

**quickly. Of course, their design could make use only of items that were**

**already on board the ship—deliveries were impossible. The lead engineer**

**poured the available materials onto a table and the others went to work,**

**solving their problem with cardboard, hoses, and duct tape.**

**Engineers never have all of the resources that they might like. They must**

**work with what they have.**

**Imagine your future. Which episode inspires you the most? Which model**

**would you most like to emulate? Will you know your piece better than**

**anyone else? Will you have the courage to speak up? Does the challenge**

**of solving a pressing problem with very limited supplies and time appeal**

**to you?**

**64. Describe the problem that the union-find program solved.**

**65. Draw the binary search tree that results when the first seven letters of the**

**alphabet are inserted in this order: d—b—a—c—f—e—g**

**66. Here are the values in elements 1–7 of an array: 7—5—6—4—3—1—2**

**Could this be a heap? How can you tell?**

**67. If the value of N is one billion, how much bigger is N2**

**than NlgN**

**14**

**68. What is a wide API? Give an example. Are wide APIs desirable?**

**69. Specify the API for a class that models a bag in a UML class diagram.**

**70. Distinguish two principal ways of writing a merge sort.**

You must decide whether to use an array or a linked list.

**71. Identify a major decision that must be made in the design a program that**

**executes the quick sort algorithm.**

**72. How is the shell sort algorithm related to the insertion sort?**

**73. Explain why insertion sort is usually faster than selection sort.**

**74. Describe the start of an algorithm that constructs a minimum spanning**

**tree.**

**75. When should we prefer a search tree to a hash table?**

**76. Red-black trees combine the best features of (ordinary) binary search trees**

**and 2–3 trees. What are those best features?**

**77. I shared programs with you that contained the most essential elements of**

**code that the authors of our textbook gave us. I recommended that you**

**also study algorithms and data structures by similarly minimalist writing**

**programs. What kinds of features of the authors’ programs did I leave out**

**of my examples?**

**78. I have introduced you to tools for composing, documenting, testing, and**

**presenting code. Write a short paragraph that describes your favorite and**

**persuades a classmate to try it.**

**15**