

## Group work for topic 1

*Each group will solve some of the problems below. Groups will solve different problems. Post solutions under the Topic 1 forum at the Blackboard Discussion Board.*

### A. Discussion of problems

1. Use the principle of optimality to describe finding the best batting average among players on teams in the major baseball leagues.
2. List two problems involving each of the following: arrays, graphs, sets of points on a coordinate graph.
3. Which of the following problems use solutions to which others, and in what way? String match, string search, pattern match
4. Distinguish a solution tree from a binary search tree.
5. Closest-pair and convex-hull are problems of what type?
6. Why is graph coloring an optimization problem?
7. What are the components of a graph that determine paths?
8. What kind of graph does the shortest-path problem apply to?
9. What is a subgraph that is connected, acyclic, and provides a minimum-cost subway system design?
10. What is an ongoing series of responses to inputs, queries, or requests?

### B. Problem specification

Using the language of predicate logic, give the specification for an algorithm that accepts an array  $A$  that stores a set  $S$  and

1. searches  $A$  for parameter value  $x$  returning *True* or *False*.
2. returns array  $A$  in ascending order.
3. returns the bitwise AND of  $A$ , where  $A$  is a bit vector.
4. returns the bitwise OR of  $A$ , where  $A$  is a bit vector.
5. tells whether any consecutive elements of  $A$  are the same.
6. tells whether all elements of  $A$  are the same.
7. tells whether any consecutive elements of  $A$  are the same.
8. tells whether any two elements of  $A$  are the same.
9. returns the maximum value in  $A$ .

### C. Computation Tree Logic

Write a CTL formula that applies to a given reactive system and asserts, for formulas  $\phi$  and  $\varphi$ , that on this system,

1. will never hold.
2. there is a way for  $\phi$  to become true
3. after the next state transition,  $\phi$  will hold.
4. only after  $\phi$  holds can  $\varphi$  cease to be true.
5. there is no way to get into the state  $\phi$ .
6. there is a way to get out of the state,  $\phi$ .

## Group work for topic 2

Each group will solve some of the problems below. Groups will solve different problems. Post solutions under the Topic 2 forum at the Blackboard Discussion Board.

1. By use of preconditions, postconditions, and loop invariants, prove or argue persuasively that the pseudocode below will correctly count and return the number of spaces in a string,  $s$ . Note that a loop invariant is an assertion about values of data items, not a narrative of what happens.

### Count-spaces(s)

```
n ← 0
i ← 1
while i ≤ length(s) do
    if s [ i ] = ' ' then
        n ← n + 1
    i ← i + 1
return n
```

Relate the above algorithm to the Hoare triple  $\langle \phi \rangle P \langle \psi \rangle$  and explain how you would go about proving the correctness of Count-spaces.

2. Consider the pseudocode below:

### Product (x, y)

```
result ← 0
For i ← 1 to x
    result ← result + y
Return result
```

- (a) what are an appropriate precondition, postcondition, and loop invariant?
- (b) How does your loop invariant ensure that the postcondition is valid?
3. Write an algorithm to compute the function  $y = a^b$ , for inputs  $a, b$ . For this algorithm, answer (a), (b) of Question 2.
4. Write an algorithm to compute the sum of the elements of an input array,  $A$ . For this algorithm, answer (a), (b) of Question 2.

5. (a) Express an algorithm iteratively for the factorial function;  
(b) By use of comments, show that it terminates and show partial correctness;
6. Write a version of the Quicksort *Partition* step in which first all the values less than the pivot are collected, then the pivot is appended, then the values greater than the pivot are collected. Show its correctness.
7. Show that the following pseudocode returns the reverse of array  $A$ , where *first* and *last* are subscripts:

### Reverse (A, first, last)

```
If first < last then
    swap (A[first], A[last])
    return Reverse (A, first+1, last-1)
else
    return A
```

8. Write an algorithm that tells whether an array of integers is in ascending order; prove correctness.
9. Write preconditions, postconditions, and loop invariants to argue that Which-sort, below, works correctly. This requires showing that Largest-to-right moves the largest element of  $A[1..n]$  into position  $n$  of  $A$ .

### Which-sort (A)

```
for i ← size(A) down to 2 do
    Largest-to-right (A, i)
```

### Largest-to-right (A, n)

```
largest ← A[1]
for i ← 2 to n do
    if A[i] > A [largest]
        largest ← i
    swap (A[largest], A[n])
```

10. (a) Express an algorithm iteratively for the factorial function;  
(b) By use of comments, show that it terminates and show partial correctness.

## Group work for topic 3

Each group will solve some of the problems below.  
Groups will solve different problems. Post solutions under the Topic 3 forum at the Blackboard Discussion Board.

### A. Given algorithms, find complexities

1. Discuss the complexity of this search algorithm, write it as a recurrence, and argue for its correctness, using preconditions, postconditions, and a loop invariant:

**Some-search (A, first, last, key)**

```
while first ≤ last do
  middle ← (first + last) ÷ 2
  if key < A[middle]
    last ← middle - 1
  else if key > A[middle]
    first ← middle + 1
  else return true
return false
```

2. State complexity of each algorithm below and justify your answer.

**Which-sort (A)**

```
for i ← size(A) down to 2 do
  Largest-to-right (A, i)
```

**Largest-to-right (A, n)**

```
largest ← A[1]
for i ← 2 to n do
  if A[i] > A[largest]
    largest ← i
swap (A[largest], A[n])
```

### B. Given problems, design and analyze solution algorithms

For each of the numbered problems, the solution is in five parts, (a) to (e).

- a. Express an algorithm iteratively (in pseudo-code or a programming language) to solve this problem.
- b. Write a recursive version of this algorithm.
- c. In a recurrence define the function that corresponds to the problem specification.
- d. Write a recurrence, based on (c), that defines the function that returns the running time for a sequence of size  $n$ .
- e. Solve the time recurrence, giving a one-sentence explanation.

***Problems:***

1. Find the largest element of a subsequence in an array. Signature:  $Max(A, first, last)$ , where  $A$  is an array,  $first$  and  $last$  are subscripts. Algorithm should return the largest element of the sequence from  $A[first]$  to  $A[last]$ , inclusive.;
2. The factorial function  
(Example:  $fact(4) = 4 \times 3 \times 2 = 24$ )  
(Alternatives: summation of  $1..x$ ;  
 $fibonacci(i)$ )
3. Return the numeric value of a sequence of 0's and 1's denoting a binary numeral
4. Return the smallest value in sequence of integers.
5. Find the number of occurrences of a string,  $s$ , in a text file,  $t$ .
6. Tell whether an array is in ascending order.

## Group work for topic 4

*Each group will solve some of the problems below. Groups will solve different problems. Post solutions under the Topic 4 forum at the Blackboard Discussion Board.*

Write a brute-force algorithm to

1. solve the closest-pair problem. Analyze.
2. solve the convex-hull problem. Analyze.
3. find the minimum spanning tree of a graph. Analyze.
4. solve the satisfiability problem. Analyze.
5. find a Hamiltonian path. Analyze.
6. solve the Traveling Salesperson problem. Analyze.
7. solve the knapsack problem. Analyze.
8. tell whether a given destination vertex in a graph is reachable from a given source vertex. Analyze.
9. solve the independent-set problem. Analyze.
10. determine whether a given set of natural numbers may be partitioned into two subsets of equal sum. Analyze.

## Group work for topic 5

*Each group will solve some of the problems below. Groups will solve different problems. Post solutions under the Topic 5 forum at the Blackboard Discussion Board.*

### A. Recurrences

Write recurrences for the functions computed by the following divide-and-conquer algorithms; write time recurrences and analyze running time.

1. Binary search
2. Quicksort
3. Breadth-first search
4. Depth-first search
5. BST search
6. BST insertion
7. Order statistic
8. Merge sort
9. Convex hull
10. Closest pair

### B. Coding and performance testing

Code and performance test the algorithms in A; comparing results to the analysis provided in A.

## Group work for topic 6

*Each group will solve some of the problems below. Groups will solve different problems. Post solutions under the Topic 6 forum at the Blackboard Discussion Board.*

1. Design an algorithm to tell whether a graph is cyclic; write a recurrence to define the function computed; write a time recurrence; state complexity.
2. Do the Prim and Kruskal algorithms produce unique optimal solutions? Justify.
3. Show that the minimal spanning tree (Prim, Kruskal) is not necessarily the same as the tree of single-source shortest paths (Dijkstra).
4. Show how the optimal-substructure property applies with each greedy algorithm discussed.
5. Show how optimal-substructure property is used in continuous-knapsack problem solution.
6. Compare pseudocode in the slides and the textbook for Dijkstra's algorithm.
7. What is the running time of the Dijkstra algorithm and why?
8. What does Huffman's algorithm find, and how?
9. Describe a greedy solution to the continuous-knapsack problem.
10. Defend or refute: The optimal-substructure property guarantees that some greedy algorithm solves a given problem.

## Group work for topic 7

*Each group will solve some of the problems below. Groups will solve different problems. Post solutions under the Topic 7 forum at the Blackboard Discussion Board.*

1. Write an algorithm to solve sorting-by-counting problem (see slide).
2. Code and analyze the time complexity of Boyer-Moore search.
3. Write an algorithm to decide path of descent of B tree
4. Find and explain complexities of  $C(n, k)$  binomial coefficient algorithms that use (a) recursion (brute force) and (b) dynamic programming.
5. Describe an algorithm to find the *mode* (most frequent element) of an array. Analyze.
6. What are AVL trees used for, and how?
7. What are the running times of *heapify*, *extract-min*, and *insert* for heaps? Relate this to the data structure used.
8. In dynamic-programming algorithms, what resource defines the main overhead of using this method?
9. Write an algorithm to solve the sorting-by-counting problem (see slide).
10. Code the Boyer-Moore search and analyze its time complexity.
11. What are the space and time considerations raised by B trees?

## Group work for topic 8

*Each group will solve some of the problems below. Groups will solve different problems. Post solutions under the Topic 8 forum at the Blackboard Discussion Board.*

1. What are the complexities of these problems w.r.t. formulas  $\phi$  in propositional logic?
  - a.  $\phi$  is a tautology
  - b.  $\phi$  is satisfiable
  - c.  $\phi$  is a contradiction
  - d.  $\phi$  holds for a given set of variable assignments
2. Relate or compare the problems of evaluating formulas in propositional logic, e.g.  $p \wedge q \vee (r \wedge \neg s)$ , using truth tables, on the one hand, and determining satisfiability of formulas in prop. logic.
3. If there is an algorithm that can transform any instance of problem  $A$  to an instance of problem  $B$ , then what can be said about the relationship of  $A$  to  $B$ ?
4. Describe the correspondence between a decision problem and a language.
5. What is DTIME( $T(n)$ )? EXPTIME? P?
6. What is NP? What is NPC?
7. What is the (very short) name of the set of problems that are decidable in time that is a polynomial function of the input size?
8. Suppose someone found an  $O(n^4)$  solution to the Traveling Salesperson problem.
  - (a) What implications would this have for other problems reducible to TSP in polynomial time?
  - (b) What implications would it have for other problems to which TSP is reducible in polynomial time?
9. Give reasons why it makes sense to associate  $P$  with tractability, and give reasons why this association has limited value.
10. What bound on what resource is given by the minimum height of a decision tree representing execution of an algorithm

## Group work for topic 9

*Each group will solve some of the problems below. Groups will solve different problems. Post solutions under the Topic 9 forum at the Blackboard Discussion Board.*

1. When are heuristics used?
2. Can the accuracy of heuristics be quantified?
3. Discuss the problem of finding a maximum matching in a bipartite graph, and one solution.
4. Why is the algorithm shown for approximately solving the stable-marriage problem  $O(n^2)$ ?
5. What does searching state spaces have to do with solving problems?
6. When is the generate-and-test algorithm unlikely to find an optimal solution?
7. When is the exhaustive search for a path through a tree likely to take a huge amount of time?
8. Why does hill climbing often find approximate rather than exact optima?
9. Compare backtracking to branch-and-bound.
10. What is the complexity of minimax, relative to the number of moves ahead it looks in a game?

## Group work for topic 10

*Each group will solve some of the problems below. Groups will solve different problems. Post solutions under the Topic 10 forum at the Blackboard Discussion Board.*

Write a parallel algorithm to find the following for an array of  $n$  integers; state complexity and justify

1. the sum
2. the maximum value
3. the OR, if the integers are all 0 or 1
4. the AND if integers are all 0 or 1
5. the minimum value
6. the number of 0's
7. the product

## Group work for topic 11

*Each group will solve some of the problems below. Groups will solve different problems. Post solutions under the Topic 11 forum at the Blackboard Discussion Board.*

1. What sort of computation is associated with computing functions, and what sort of computation is associated with providing services?
2. Distinguish two types of interaction by the number of active agents involved.
3. Distinguish two types of interaction by the use or non-use of the environment in interaction.
4. Whereas algorithmic computation operates on strings and natural numbers, what does sequential interaction operate on?
5. What measures of efficiency are applied in interactive computing?
6. In multi-agent systems, what scalability issues are raised by the choice between message passing and shared access to storage?
7. Argue that interaction is more powerful than algorithms.
8. Argue that interaction is not more powerful than algorithms.
9. Argue that multi-stream interaction is more powerful than sequential interaction; that it is not more powerful.
10. Summarize Peter Wegner's view of reasoning versus interaction.