Purdue University
West Lafayette
CS 182

Ahmed Mahmood
Summer 2017

Assignment 2 Solutions

Due: Tuesday, June 27, 2017, upload before 11:30pm

1) (10 pts.) Do Exercise 36 of Section 1.5 (page 68).

Graded by : Sneha Balasubramanian

Solution:

a) Let A(x) be "x has lost more than one thousand dollars playing the lottery"

$\forall$x $\neg$A(x)

the negation is :

$\neg$ $\forall$(x)$\neg$A(x)

=$\exists$(x)$\neg(\neg$A(x))

=$\exists$(x)A(x) this means: there is someone who has lost more than a thousand dollars playing the lottery.

b) Let the universe be students in this class and let $B(x, y)$ be the statement "x has chatted with y". The given statement is $\exists x \exists y (B(x, y) \wedge \forall z (B(x, z) \rightarrow (z = y)))$. The negation of the given statement is $\forall x \forall y (\neg B(x, y) \vee \exists z (B(x, y)$ / or "all students in this class have either chatted with no other students or have chatted with at least two students".

c) Let B(x,y) be "x has sent mail to y"

$\forall$x$\forall$y$\forall$z $\neg$ (x $\neq$ y $\wedge$ c $\neq$ z $\wedge$ y $\neq$ z $\rightarrow$ B(x,y) $\wedge$ B(x,z) $\wedge$ $\neg$ $\exists$a[a $\neq$ y $\wedge$ a $\neq$ z $\wedge$ B(x,a)])

the negation is :

$\neg$ $\forall$x$\forall$y$\forall$z $\neg$ (x $\neq$ y $\wedge$ c $\neq$ z $\wedge$ y $\neq$ z $\rightarrow$ B(x,y) $\wedge$ B(x,z) $\wedge$ $\neg$ $\exists$a[a $\neq$ y $\wedge$ a $\neq$ z $\wedge$ B(x,a)])

=$\exists$x $\neg$ $\forall$y$\forall$z $\neg$ (x $\neq$ y $\wedge$ c $\neq$ z $\wedge$ y $\neq$ z $\rightarrow$ B(x,y) $\wedge$ B(x,z) $\wedge$ $\neg$ $\exists$a[a $\neq$ y $\wedge$ a $\neq$ z $\wedge$ B(x,a)])

=$\exists$x $\exists$y $\neg$ $\forall$z $\neg$ (x $\neq$ y $\wedge$ c $\neq$ z $\wedge$ y $\neq$ z $\rightarrow$ B(x,y) $\wedge$ B(x,z) $\wedge$ $\neg$ $\exists$a[a $\neq$ y $\wedge$ a $\neq$ z $\wedge$ B(x,a)])

=$\exists$x $\exists$y $\exists$z $\neg$ $\neg$ (x $\neq$ y $\wedge$ c $\neq$ z $\wedge$ y $\neq$ z $\rightarrow$ B(x,y) $\wedge$ B(x,z) $\wedge$ $\neg$ $\exists$a[a $\neq$ y $\wedge$ a $\neq$ z $\wedge$ B(x,a)])

=$\exists$x $\exists$y $\exists$z (x $\neq$ y $\wedge$ c $\neq$ z $\wedge$ y $\neq$ z $\rightarrow$ B(x,y) $\wedge$ B(x,z) $\wedge$ $\neg$ $\exists$a[a $\neq$ y $\wedge$ a $\neq$ z $\wedge$ B(x,a)])

which means there is a student in the class that has sent a mail to exactly two other students in the class

Another solution:

The given statement is

$$\neg\exists x\exists y\exists z\left(B\left(x,y\right)\wedge B\left(x,z\right)\wedge\forall w\left(B\left(x,w\right)\rightarrow\left(\left(w=y\right)\vee\left(w=z\right)\right)\right)\right).$$

The negation of the given statement is

$$\exists x\exists y\exists z\left(B\left(x,y\right)\wedge B\left(x,z\right)\wedge\forall w\left(B\left(x,w\right)\rightarrow\left(\left(w=y\right)\vee\left(w=z\right)\right)\right)\right),$$

or "there is a student in this class who has e-mailed exactly two other students".

d) Let D(x,y) be "student x has solved exercise y of the book"

$\exists x \forall y D(x,y)$
the negation is :
$\neg \exists x \forall y D(x,y)$
$= \forall x \neg \forall y D(x,y)$
$= \forall x \exists y \neg D(x,y)$
which means no student has solved every exercise in the book.

e) Let $E(x,y)$ be the statement "x has solved exercise y" and $F(y,z)$ be the statement "exercise y is in section z". The given statement is $\forall x \exists z \forall y (\neg E(x,y) \vee \neg F(y,z))$. The negation of the given statement is $\exists x \forall z \exists y (E(x,y) \wedge F(y,z))$, or "there is a student who has solved at least one exercise in every section of the book".

2) (10 pts.) Do Exercise 34 of Section 1.6 (page 80).
Graded by : Sneha Balasubramanian
Solution:
1) p: logic is difficult
q: many students like logic
r: mathematics is easy

2) $p \vee (\neg q)$
$r \rightarrow \neg p$

a) Translate into symbols , given $q \rightarrow \neg r$
If q is true, p is true, in turn r is false or $\neg r$ is true, then the statement is true. if q is not true, then the statement is true in this case as well.
Result: Valid

b) Translate into symbols , given $\neg r \rightarrow \neg q$
if $\neg r$ is true, then r is false and we cannot conclude anything beyond this.
Result: Not Valid.

c) Translate into symbols , given $\neg r \vee p$
If p is false, then p is true and r is true , in turn making $\neg r$ false. this means statement cannot be true.
Result: Not Valid

d) Translate into symbols , given $\neg p \vee \neg r$
the second assumption is equal to the given statement
Result: Valid

e) Translate into symbols , given $\neg q \rightarrow (\neg r \vee \neg p)$
if $\neg q$ is true, then $(\neg r \vee \neg p)$ is true. If $\neg q$ is false, then the statement is always true.
Result: Valid.

3) (10 pts.) Do Exercise 18 of Section 1.7 (page 91).
Graded by : Sneha Balasubramanian
Solution:
a)
Proof by contraposition :
The contrapositive statement is "if n is odd then 3n+2 is odd". thus we need to assume n is odd. By the definition of odd numbers, there is an integer k such that
n= 2k + 1.

Subsitituting n=2k+1 in 3n+2, we get
3n + 2 = 3(2k + 1) + 2 = 6k + 3 + 2 = 6k + 4 + 1 = 2(3k+2) + 1.
thus we can find an integer l=3k + 2 such that
3n + 2 = 3l + 1.
this means 3n + 2 is odd.
since the contrapositive is true, the statement is also true.

b)
Proof by contradiction :
Suppose n is not even and 3n + 2 is even. Since n is not even, it must be odd. By the definition of odd numbers, there is an integer k such that
n= 2k + 1.

Subsitituting n=2k+1 in 3n+2, we get
3n + 2 = 3(2k + 1) + 2 = 6k + 3 + 2 = 6k + 4 + 1 = 2(3k+2) + 1.
thus we can find an integer l=3k + 2 such that
3n + 2 = 3l + 1.
this means 3n + 2 is odd.
Up till now, our assumption "n is odd" leads to the contradiction that 3n + 2 is both even and odd. This is a contradiction. This means "n is odd" is a false statement. thus n must be even.

4) (10 pts.) Do Exercises 2 and 4 of Section 1.8 (page 108).
Graded by : Ramya Vulimiri

2) We have to show that there exist no positive integers x,y,z, such that $x^3 + y^3 = z^3$ where $z^3 < 1000$.

Proof by exhaustion:
There are 9 perfect cubes less than 1000 - $1^3, ...9^3 = 729$. You can check in various ways that the sums of different combinations of these cubes doesn't ever lead to a perfect cube.
One method could be tabulating and summing all combinations of cubes, with $1^3 = 1, 2^3 = 8, 3^3 = 27, 4^3 = 64, 5^3 = 125, 6^3 = 216, 7^3 = 343, 8^3 = 512, 9^3 = 729$ as rows and columns and the cell values being the sum of corresponding row and column and check that none of the cell values is a perfect cube.
Another method could be the rows and columns being the perfect cubes of 1,...,9 and the cell values being differ-

ence of the cubes, considered only when greater than 0.

4) We have to show that min(a, min(b, c)) = min(min(a, b), c) whenever a, b, and c are real numbers.

There are three main cases for this problem depending on which of a, b, and c are the smallest. (Alternatively, if you list all combinations, there will be 6 cases.)

Case (i): a is the smallest of the three. Then, LHS will be equal to a since $a \leq min(b, c)$, regardless of which is the min of b and c. RHS = min(min(a, b), c) = min(a, c) = a.

Case (ii): b is the smallest of the three. Then, LHS = min(a, min(b, c)) = min(a,b) = b. RHS = min(min(a, b), c) = min(b, c) = b.

Case (iii): c is the smallest of the three. Then, LHS = min(a, min(b, c)) = min(a,c) = c. RHS = min(min(a, b), c) = c.

5) (20 pts.) Given is an array A of size n containing integers in arbitrary order; n is even. For each of the problems described below describe and analyze an efficient algorithm.

Explain your solution in a pseudo code. Explain the achieved running times in big-O. Include a brief description on the correctness of your algorithm.

Graded by : Ramya Vulimiri

1. MaxPart: Partition the n integers in array A into two sets $S1$ and $S2$, each of size $n/2$, such that the difference between the sum of the elements in $S1$ and the sum of the elements in $S2$ is a maximum.

   Solution: The difference between the sum of the elements in S1 and the sum of elements in S2 is maximum when you have the n/2 least elements in one set and the remaining in the other.

   Correctness: We want to prove that maximum difference is achieved in the above case. This can be checked by a simple proof by contradiction. Consider sets S1 and S2, where we claim to have max difference δ but you do not have all the least elements in S1 - say you have elements $a_1, .., a_{i-1}, a_i$ in S1 and $a_j, .., .., a_n$ in S2 where $a_1, .., a_{i-1}, a_j$ are the smallest elements and you have $a_i > a_j$. We can see that difference can be increased by exchanging a_i and a_j. This is a contradiction to the claim that $a_1, .., a_{i-1}, a_i$ in S1 and $a_j, .., .., a_n$ in S2 was the partition with maximum difference.

   You can use merge sort to sort the elements and then the first half of the array can go into S1 and the second half into S2. Merge Sort is O(n *log* n) and iterating through the elements to put in S1 and S2 will be O(n). Overall, the complexity will be O(n *log* n + n) = O(n *log* n).

---

procedure MaxPart(A: Array of integers)
  Array S_1, S_2 ▷ Empty arrays of size $n/2$
  MergeSort(A)
  for i=1 ..$n/2$ **do** ▷ indices starting from 1
    $S_1[i] = A[i]$
  for i=($n/2 + 1$) .. n **do**
    $S_2[i] = A[i]$
  return S_1, S_2

---

2. MinDiff: Find the minimum pairwise difference δ for the elements in array A. It is defined as the minimum so that $|A[i] - A[j]| \geq \delta$ for all i and j.

Solution: Sort the elements in the array and find differences between adjacent elements to choose the minimum pairwise difference(this reduces the number of comparisons - you do not need to compare every element with every other).

Correctness: If we were finding the minimum pairwise difference by brute force, we would compute distances $|A[i] - A[j]|$ for all i and j and then find the minimum value from them. But we can reduce the number of computations needed by observing that if we have the elements in an ascending order, we need only compare distances between adjacent elements since the distances between non-adjacent elements will be greater than or equal to distances between adjacent elements and we want to find the minimum pairwise distance.

Merge Sort is O($n\ log\ n$) and scanning the elements of the array to find differences will be O(n). Overall, the complexity will be O($n\ log\ n + n$) = O($n\ log\ n$).

---

procedure MinDiff(A: Array of integers)
    MergeSort(A)
    $\delta = \infty$
    for $i = 1$ .. $n - 1$ **do** ▷ indices starting from 1
        if $|A[i] - A[i + 1]| \leq \delta$
            $\delta = |A[i] - A[i + 1]|$
    return δ

[Note: There can be many different ways to solve the above two problems.]

Solution 6: Graded by Rashmi Soni

Arranging the functions in order such that each function is big-O of the next function:

$19logn$, $\sqrt{n}$, n^2logn, $\frac{n^3}{10^5}$, 3.2^n, 2.3^n, $3n!$, $14n^n$

Proof for consecutive functions:

1) $19logn$ is $O(\sqrt{n})$

$logn < \sqrt{n} \quad \forall n \geq 1$

$19logn < 19\sqrt{n} \quad \forall n \geq 1$

So, C=19, K=1

2) $\sqrt{n}$ is $O(n^2logn)$

$\sqrt{n} \leq n^2 \quad \forall n > 1$

$\sqrt{n} \leq n^2logn \quad \forall n > 2.1$

So, C=1, K=3

3) n^2logn is $O(\frac{n^3}{10^5})$

$logn \leq n \quad \forall n > 1$

$n^2logn \leq n^3 \quad \forall n > 1$

$n^2logn \leq \frac{n^3}{10^5} \quad \forall n > 1$

So, C=1, K=2

4) $\frac{n^3}{10^5}$ is $O(3.2^n)$

$n^3 < 2^n \quad \forall n > 1$

$\frac{n^3}{10^5} < \frac{2^n}{10^5} \quad \forall n > 10$

$\frac{n^3}{10^5} < (3/3)\frac{2^n}{10^5} \quad \forall n > 10$

So, C=$\frac{1}{3.10^5}$, K=12

5) 3.2^n is $O(2.3^n)$

$2^n \leq 3^n \quad \forall n > 0$

$3.2^n \leq 3.3^n \quad \forall n > 0$

$3.2^n \leq (3/2).2.3^n \quad \forall n > 0$

So, C=1.5, K=1

6) 2.3^n is $O(3n!)$

$3^n \leq n! \quad \forall n > 7$

$2.3^n \leq 2n! \quad \forall n > 7$

$2.3^n \leq (2/3)3n! \quad \forall n > 7$

So, C=2/3, K=8

7) $3n!$ is $O(14n^n)$

$n! \leq n^n \quad \forall n \geq 1$

$3n! \leq 3n^n$

$3n! \leq (3/14)14n^n \quad \forall n \geq 1$

So, C=3/14, K=1

Solution 7: Graded by Rashmi Soni

Given, max distance covered through car tank without stopping = d miles

number of gas stations = n

Distance from gas station i to the subsequent gas station (i+1) is Dist[i]; where the array Dist[] holds (n-1) values

We propose the following greedy algorithm for this problem: Firstly, the car starts out from Loc1 with the max possible gas tank capacity. Then, it calculates the position of the gas station that is as far as possible from car's current position and within the reach of d miles.It stops at this gas station; thereby refilling the tank full. It again starts out and calculates the position of the next gas station that is farthest from the current gas station and within the reach of d miles. This process is continued until it reaches Loc2.

The pseudo-code for the above algorithm is given as :

Algorithm *MinStops* (d,Dist[])
begin
list_GS = []
covered_distance = 0
i = 1
while (i ≤ (n-1))
    if (*covered_distance* + *Dist*[*i*]) ≤ d
        covered_distance = *covered_distance* + Dist[i]
        i = i + 1
    else
        //make stop at G_i, so add G_i to the output gas stations list
        append G_i to *list_GS*
        covered_distance = 0
return *list_GS*
end

The complexity of the algorithm is O(n).