The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 4 MODULE, AUTUMN 2011–2012

ALGORITHM DESIGN

Time allowed 3 hours

Candidates must NOT start writing their answers until told to do so.

Model Solutions

1 a) Applying the assignment axiom and conditional rule, the assertions are expanded as shown below.

$\{ \quad M{<}N \ \wedge\ \mathsf{f}(M){\le}0{\le}\mathsf{f}(N) \quad \}$

$\{ \quad M{\le}M{<}N{\le}N \ \wedge\ \mathsf{f}(M){\le}0{\le}\mathsf{f}(N) \quad \}$

$i,j := M,N$;

$\{$ **Invariant:** $M{\le}i{<}j{\le}N \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(j) \quad \}$

do $i{<}j \ \rightarrow$ $\{ \quad i{+}1{\neq}j \ \wedge\ i{<}j \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(j) \quad \}$

$\{ \quad i{<}(i{+}j){\div}2{<}j \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(j) \quad \}$

$k := (i{+}j){\div}2$;

$\{ \quad i{<}k{<}j \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(j) \quad \}$

if $\mathsf{f}(k){\le}0 \ \rightarrow$ $\{ \quad i{<}k{<}j \wedge \mathsf{f}(i){\le}0{\le}\mathsf{f}(j) \wedge \mathsf{f}(k){\le}0 \quad \}$

$\{ \quad k{<}j \ \wedge\ \mathsf{f}(k){\le}0{\le}\mathsf{f}(j) \quad \}$

$i := k$

$\{ \quad i{<}j \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(j) \quad \}$

$\Box \quad 0{\le}\mathsf{f}(k) \rightarrow$ $\{ \quad i{<}k{<}j \wedge \mathsf{f}(i){\le}0{\le}\mathsf{f}(j) \wedge 0{\le}\mathsf{f}(k) \quad \}$

$\{ \quad i{<}k \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(k) \quad \}$

$j := k$

$\{ \quad i{<}j \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(j) \quad \}$

fi

$\{ \quad i{<}j \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(j) \quad \}$

od

$\{ \quad M{\le}i{<}j{\le}N \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(j) \ \wedge\ i{+}1{=}j \quad \}$

$\{ \quad M{\le}i{<}N \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(i{+}1) \quad \}$.

From this, we read off the verification condition for the initialisation:

$[\ M{<}N \ \wedge\ \mathsf{f}(M){\le}0{\le}\mathsf{f}(N) \ \Rightarrow\ M{\le}M{<}N{\le}N \ \wedge\ \mathsf{f}(M){\le}0{\le}\mathsf{f}(N)\ ]$.

and the verification conditions for the conditional correcness of the inner loop:

$$[\ i{+}1{\neq}j \ \wedge\ i{<}j \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(j) \ \Rightarrow\ i{<}(i{+}j){\div}2{<}j \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(j)\ ]$$

and

$$[\ i{<}k{<}j \wedge \mathsf{f}(i){\le}0{\le}\mathsf{f}(j) \wedge \mathsf{f}(k){\le}0 \ \Rightarrow\ k{<}j \ \wedge\ \mathsf{f}(k){\le}0{\le}\mathsf{f}(j)\ ]$$

and

$$[\ i{<}k{<}j \wedge \mathsf{f}(i){\le}0{\le}\mathsf{f}(j) \wedge 0{\le}\mathsf{f}(k) \ \Rightarrow\ i{<}k \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(k)\ ]$$.

Finally, we have the verification condition for termination:

$$[\ M{\le}i{<}j{\le}N \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(j) \ \wedge\ i{+}1{=}j \ \Rightarrow\ M{\le}i{<}N \ \wedge\ \mathsf{f}(i){\le}0{\le}\mathsf{f}(i{+}1)\ ]$$.

(Marking scheme: 5 marks for each vc. If the program annotation is given, this will be used to reward students who have made mistakes. The program annotation is not specified in the question so just listing the verification conditions correctly is fine. However if this is done and the verification conditions

are incorrect it may lead to a mark of 0 being awarded for that vc.)

2 (a) The array elements are reversed from the outside inwards. The two indices j and k delimit that part of the array still to be reversed. N is the length of the array. A_0 is a ghost variable used to relate the value of the array a to its initial value.

$\{ \quad 0 \le N \ \wedge \ \langle \forall i \ : \ 0 \le i < N \ : \ a[i] = a_0[i] \rangle \quad \}$

$j,k := 0,N$

$\{$ **Invariant:**

$0 \le j \le N \ \wedge \ 0 \le k \le N \ \wedge \ j+k = N$

$\wedge \quad \langle \forall i \ : \ 0 \le i < j \ \vee \ k \le i < N \ : \ a[i] = a_0[N-1-i] \rangle$

$\wedge \quad \langle \forall i \ : \ j \le i < k \ : \ a[i] = a_0[i] \rangle$

Bound function: $k-j \quad \}$

; do $j < k \ \rightarrow \quad swap(j\,,k-1)$; $j,k := j+1\,,k-1$

od

$\{ \quad \langle \forall i \ : \ 0 \le i < N \ : \ a[i] = a_0[N-1-i] \rangle \quad \}$

Note the use of two variables j and k. This preserves the symmetry between the top and bottom halves of the array and helps to avoid error in the calculation of the array indices.

(This question seems very straightforward but my experience is that it is easy to make a mistake. A common mistake is to assert that $j \le k$, and to terminate the loop when j and k are equal.)

(b)

$\{ \quad 0 \le N \ \wedge \ \langle \forall i \ : \ 0 \le i < N \ : \ a[i] = a_0[i] \rangle \quad \}$

$j,k,c := 0,N,0$

$\{$ **Invariant:**

$0 \le j < N \ \wedge \ 0 \le k < N \ \wedge \ j+k = N$

$\wedge \quad \langle \forall i \ : \ 0 \le i < j \ \vee \ k \le i < N \ : \ a[i] = a_0[N-1-i] \rangle$

$\wedge \quad \langle \forall i \ : \ j \le i < k \ : \ a[i] = a_0[i] \rangle$

$\wedge \quad c = \langle \Sigma\, i,i' \ : \ 0 \le i < i' \le N \wedge i+i' = N \wedge a_0[i] \ne a_0[i'-1] \ : \ 1 \rangle$

Bound function: $k-j \quad \}$

; do $j < k \ \rightarrow$ if $a[j] \ne a[k-1] \rightarrow c := c+1$

$\square \ \ a[j] = a[k-1] \rightarrow$ skip

fi ;

$swap(j\,,k-1)$;

$j,k := j+1\,,k-1$

od

$\{ \quad \langle \forall i \ : \ 0 \le i < N \ : \ a[i] = a_0[N-1-i] \rangle$

$\wedge \quad 2\times c = \langle \Sigma i : 0 \le i < N \wedge a_0[i] \neq a_0[N-1-i] : 1 \rangle \quad \}$

Marking scheme. Initialisation of c and addition of conditional statement: 5. Augmentation of invariant and postcondition: 10.

3 (a) As pointed out, the function mapping k and l to $M\times k - N\times l$ is strictly increasing in k and strictly decreasing in l. A saddleback search (discussed in the lectures) can be used to determine how often this function has the value 0. We introduce variables k, l and count, with the invariant property:

$$0 \le k \le N \ \wedge \ 0 \le l \le M \quad \wedge \quad \mathsf{count} + S.(k,l) \ = \ S.(0,0)$$

where

$$S.(k,l) \ = \ \langle \Sigma\, i,j \ : \ k \le i \le N \ \wedge \ l \le j \le M \ \wedge \ M\times i = N\times j \ : \ 1 \rangle \quad .$$

The initialisation is straightforward: set all of k, l and count to 0. For the loop body, we observe that, because $M\times k - N\times l$ is strictly increasing in k, if $M\times k < N\times l$, k can be incremented by 1. Conversely, because $M\times k - N\times l$ is strictly decreasing in l, if $N\times l < M\times k$, l can be incremented by 1. When $M\times k$ and $N\times l$ are equal, the count is incremented as well as both k and l. The loop is terminated when either k equals N or l equals M. On termination of the loop,

$$(k = N \ \vee \ l = M) \quad \wedge \quad \mathsf{count} + S.(k,l) \ = \ S.(0,0) \quad .$$

Since M is the unique solution of the equation $j{::}\ M\times N = N\times j$, and N is the unique solution of the equation $i{::}\ M\times i = N\times M$, we conclude that $S.(N,l) = S.(k,M) = 1$ (when $l \le M$ and $k \le N$). That is, on termination,

$$\mathsf{count}+1 = S.(0,0) \quad .$$

We thus obtain the following algorithm.

$\{ \quad 0 \le M \ \wedge \ 0 \le N \quad \}$

$k, l, \mathsf{count} \ := \ 0, 0, 0 \ ;$

$\{$ **Invariant:** as above $\}$

do $k \neq N \wedge l \neq M \ \rightarrow$ if $M\times k < N\times l \rightarrow k := k+1$

$\square \quad N\times l < M\times k \rightarrow l := l+1$

$\square \quad M\times k = N\times l \ \rightarrow \ k, l, \mathsf{count} \ := \ k+1, l+1, \mathsf{count}+1$

fi

od ;

count := count+1

$\{ \quad \mathsf{count} = S.(0,0) = \langle \Sigma\, i,j \ : \ 0 \le i \le N \ \wedge \ 0 \le j \le M \ \wedge \ M\times i = N\times j \ : \ 1 \rangle \quad \} \quad .$

The measure of progress is $(M-l)+(N-k)$. This is bounded below by 0 (because $0 \le k \le N \ \wedge \ 0 \le l \le M$) and is decreased at each iteration (either by 1 or by 2).

(b) In order to avoid the repeated multiplication of M by k and N by l, we maintain a variable d with the invariant property $d = M\times k - N\times l$. The modifications are straightforward:

$\{ \quad 0 \leq M \ \wedge \ 0 \leq N \quad \}$

$k, l, \mathsf{count}, d \ := \ 0, 0, 0, 0 \ ;$

$\{$ **Invariant:** as above $\}$

do $k \neq N \wedge l \neq M \rightarrow$ if $d < 0 \ \rightarrow \ k, d \ := \ k{+}1, d{+}M$

$\square \quad d > 0 \ \rightarrow \ l, d \ := \ l{+}1, d{-}N$

$\square \quad d = 0 \ \rightarrow \ k, l, \mathsf{count}, d \ := \ k{+}1 \ , \ l{+}1 \ , \ \mathsf{count}{+}1, d{+}M{-}N$

fi

od ;

count := count+1

$\{ \quad \mathsf{count} = S.(0,0) = \langle \Sigma i,j \ : \ 0 \leq i \leq N \ \wedge \ 0 \leq j \leq M \ \wedge \ M \times i = N \times j \ : \ 1 \rangle \quad \} \quad .$

(Alternative solutions are, of course, possible. A clear justification must be given that progress is made. That is, the measure of progress must be given together with the bounds on k and l .)

4 a) [Bookwork] The invariant states that the set of nodes reachable from s is the set of black nodes together with the set of nodes reachable from a grey nodes. Effectively $black$ accumulates reached nodes whilst $\{s\}$ is replaced by $grey$.

b) [Bookwork] A model solution will begin by providing a table showing the sets, the operations on the sets and the frequency with which the operations are executed. The choice of data structure is based on an efficient execution of each operation, taking account of the frequency with which the operation is executed.

Set	Operation	Frequency
$black$	$:= \ \emptyset$	1
	add element	$\|N\|$
$grey$	$:= \ \{s\}$	1
	$\neq \emptyset$	$\|N\|$
	choose and remove	$\|N\|$
	add element	$\|N\|$
$white$	$v \in$	$\|E\|$
	remove element	$\|N\|$

Table 4.1: Sets. Operations and Frequency. (N denotes the set of nodes, and E denotes the set of edges.)

It is not necessary to implement all three of the sets (because of the invariant that N is partitioned by all three sets). The set $white$ is implemented by a boolean array indexed by nodes, and the set $grey$ is implemented by a stack (depth-first search) or a queue (breadth-first search). The use of a boolean array is problematic if instead of determining all the nodes reachable from s it is required to determine whether a given node is reachable from s and the graph is very large (such as the game graph for a game like chess).

c) The following solution specialises Wagner's algorithm. A solution based on Dijkstra's algorithm is acceptable. Using a queue is acceptable (but they must get the distance calculations right!)

Wagner's solution splits $grey$ into two stacks, $grey0$ and $grey1$. Nodes are chosen from $grey0$ and added to $grey1$. When $grey0$ becomes empty, it is swapped with $grey1$. An additional variable k records the distance of nodes in $grey0$ from the start node.

{ $f \in$ reachable.$\{s\}$ }

$black, grey0, grey1, white := \emptyset, \{s\}, \emptyset, N-\{s\}$; $distance[s], k := 0,0$;

{ **Invariant:** See below

Bound function: $|white|$ }

while $f \notin black$ **do**

begin

if $grey0 = \emptyset$ **then** $grey0, grey1, k := grey1, \emptyset, k+1$;

$u := fst.grey0$; add u to $black$, remove it from $grey0$;

for $v \in$ directlyreachable.u

do if $v \in white$

then add v to $grey1$; remove it from $white$;

$distance[v] := k+1$; $predecessor[v] := u$

end

{ A shortest path to f has length $distance[f]$ }

The invariant property has three parts: (a) for all nodes u in $black$, $distance[u]$ is the length of a shortest edge-count path to u and for $u \neq s$, $predecessor[u]$ is the predecessor of u on such a path, (b) for all nodes u in $grey0$, k is the length of a shortest edge-count path to u, and (c) for all nodes v in $white$, if there is a path from s to v then the shortest edge-count path to v has length either k plus the length of a shortest edge-count path from a node u in $grey0$ to v or $k+1$ plus the length of a shortest edge-count path from a node u in $grey1$ to v.

5 See the graph in fig. 5.1. (The "0"s labelling the upper edges should be read as $\emptyset$.) Note that it is easier to see how the nodes and edges in the graph relate to the inductive definition of nc by rewriting S nc k as $\langle \exists T : S = T \cup \emptyset : T \text{ nc } k \rangle$. In this way, the given formula becomes

$$[\ S \text{ nc } k+1 \ \equiv\ \langle \exists T : S = T \cup \emptyset : T \text{ nc } k \rangle \ \vee\ \langle \exists T : S = T \cup \{k\} : T \text{ nc } k-1 \rangle\ ] \quad .$$

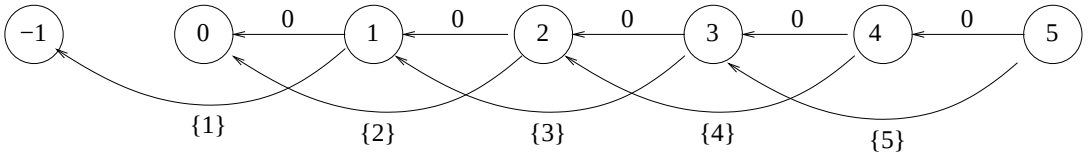

Figure 5.1 Representing the solution space.

The calculation of opt.$(k+1)$ is as follows.

opt.$(k+1)$

$=$ { definition }

$\langle \Uparrow S : S \text{ nc } k{+}1 : \text{sum}.S \rangle$

$=$ { supplied formula }

$\langle \Uparrow S \ : \ S \text{ nc } k \ \vee \ \langle \exists T : S = T \cup \{k\} : T \text{ nc } k{-}1 \rangle \ : \ \text{sum}.S \rangle$

$=$ { range disjunction (and trading),
1st component: definition of opt }

$\text{opt}.k \uparrow \langle \Uparrow T : T \text{ nc } k{-}1 : \langle \Uparrow S : S = T \cup \{k\} : \text{sum}.S \rangle \rangle$

$=$ { one-point rule }

$\text{opt}.k \uparrow \langle \Uparrow T : T \text{ nc } k{-}1 : \text{sum}.(T \cup \{k\}) \rangle$

$=$ { $\langle \Uparrow T : T \text{ nc } k{-}1 : \text{sum}.(T \cup \{k\}) \rangle$

$=$ { range splitting (noting that $[\ T \text{ nc } k{-}1 \Rightarrow k \notin T\ ]$),
one-point rule and definition of sum }

$\langle \Uparrow T : T \text{ nc } k{-}1 : \text{sum}.T + A[k] \rangle$

$=$ { distributivity of addition over max
(the "principle of optimality") }

$\langle \Uparrow T : T \text{ nc } k{-}1 : \text{sum}.T \rangle \ + A[k]$

$=$ { definition }

$\text{opt}.(k{-}1) \ + \ A[k]$ }

$\text{opt}.k \uparrow (\text{opt}.(k{-}1) \ + \ A[k])$.

Suppose $N = -1 \vee N = 0$. Then

$\text{opt}.N$

$=$ { definition }

$\langle \Uparrow S \ : \ S \text{ nc } N \ : \ \text{sum}.S \rangle$

$=$ { $[\ N = -1 \vee N = 0 \ \Rightarrow \ (S \text{ nc } N \equiv S = \emptyset)\ ]$ }

$\langle \Uparrow S : S = \emptyset : \text{sum}.S \rangle$

$=$ { one-point rule, $\text{sum}.\emptyset = 0$ }

0 .

Fig. 5.2 illustrates the general structure of the graph. The length of a path is the sum of the labels of the edges forming the path. Calculation of opt.k for a given value of k is equivalent to finding a longest terminating path in the graph starting at the node labelled k.

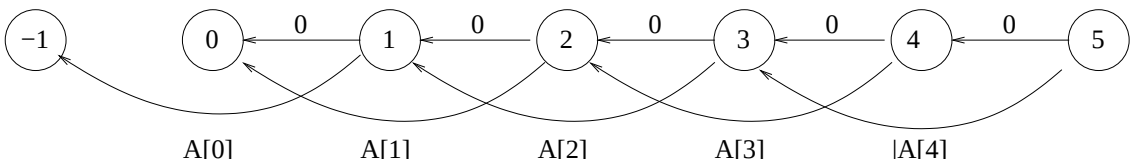

Figure 5.2 Longest-path Problem