Name: Matriculation Number:

General CS II (320201) Final Exam
May 27 2009

You have two hours(sharp) for the test;
Write the solutions to the sheet.

The estimated time for solving this exam is 104 minutes, leaving you 16 minutes for revising your exam.

You can reach 104 points if you solve all problems. You will only need 100 points for a perfect score, i.e. 4 points are bonus points.

Different problems test different skills and knowledge, so do not get stuck on one problem.

	To be used for grading, do not write here												
prob.	1.1	2.1	3.1	4.1	5.1	5.2	5.3	6.1	6.2	6.3	7.1	Sum	grade
total	4	6	8	15	15	8	6	20	8	6	8	104	
reached													

Please consider the following rules; otherwise you may lose points:

- "Prove or refute" means: If you think that the statement is correct, give a formal proof. If not, give a counter-example that makes it fail.
- Always justify your statements. Unless you are explicitly allowed to, do not just answer "yes" or "no", but instead prove your statement or refer to an appropriate definition or theorem from the lecture.
- If you write program code, give comments!

1 An Old GenCS Favourite

Problem 1.1 (Function Definition) 4pt 4min

Let A and B be sets. State the definition of the concept of a partial function with domain A and codomain B. Also state the definition of a total function with domain A and codomain B.

Solution: Let A and B be sets, then a relation $R \subseteq AB$ is called a **partial/total function**, iff for each $a \in A$, there is at most/exactly one $b \in B$, such that $\langle a, b\rangle \in R$.

2 Graphs

Problem 2.1 (Graphs and trees)

6pt
6min

Given the graph G below:

$$G = \langle \{a, b, c, d, e, f, g\}, \{\langle b, a\rangle, \langle a, d\rangle, \langle b, c\rangle, \langle c, d\rangle, \langle d, e\rangle, \langle e, d\rangle, \langle d, g\rangle, \langle c, f\rangle, \langle e, f\rangle\}\rangle$$

a Find a tree T that contains all the vertices of G as nodes and its set of edges is contained in the set of edges of G.

b Explain your choice of the root.

c Name a difference between graphs and trees.

Solution:

b. B since it's a initial node (source), so we can't place it in the tree as a child of something because it has no parrent.

c. I was aiming for no cycles but there are other things they can answer too.

---

3 Binary numbers

Problem 3.1 (8-bit) 8pt 8min

1. What is the largest number (in decimal) that can be represented in 8-bit two's complement?
2. What is its hexadecimal representation?
3. What is the smallest (most negative) number that can be represented in 8-bit two's complement?
4. What is this number in the "$(n+1)$-bit signed binary number system"?

Solution:

1. Largest number : 01111111, which is 127.
2. In hexadecimal: $7F$ (80 minus 1)
3. Most negative : 100000000, which is -128.
4. 128 is 1111111, thus in the signed binary system it is 11111111.

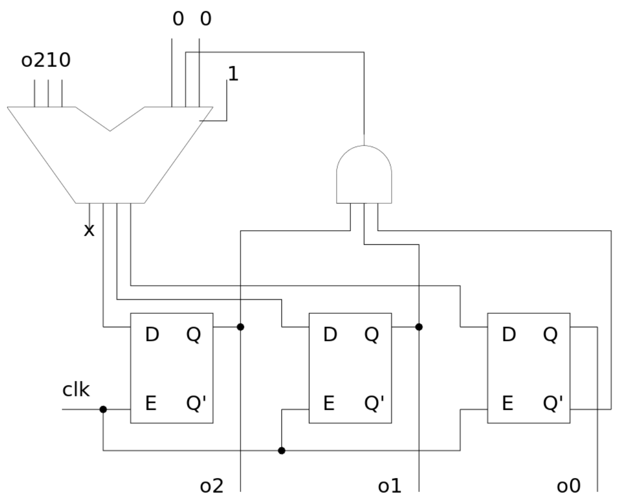
---

4 Memory

Problem 4.1 (A counter for a dice) 15pt 15min

You are asked to make the circuit for a special binary counter reflecting a 6-sided dice. The three-bit counter should count the following sequence $\ldots, 1, 2, 3, 4, 5, 6, 1, 2, 3, \ldots$. The circuit shall have three bits as output ($o2$, $o1$ and $o0$), which encode the binary numbers between 1 and 6. There is only one input signal *clk* which gives a short pulse every time the counter should increment. You can assume that your circuit is initialized in a good state. You may use all gates, flip-flops and other circuit blocks introduced in class or in the homework. You may also use constant values as input signals. Do not worry too much about timing issues but focus on the logic of your solution.

Solution:



Here is an alternative, less costly and more sophisticated solution:

Problem 0.1 (A counter for a dice)
You are asked to make the circuit for a special binary counter reflecting a 6-sided dice. The three-bit counter should count the following sequence $\ldots, 1, 2, 3, 4, 5, 6, 1, 2, 3, \ldots$. The circuit shall have three bits as output ($o2$, $o1$ and $o0$), which encode the binary numbers between 1 and 6. There is only one input signal *clk* which gives a short pulse every time the counter should increment. You can assume that your circuit is initialized in a good state. You may use all gates, flip-flops and other circuit blocks introduced in class or in the homework. You may also use constant values as input signals. Do not worry too much about timing issues but focus on the logic of your solution.

5 Machines

Problem 5.1 (FOR loop in Assembler) 15pt 15min

Translate the following piece of high level pseudocode into assembler code, where the variables n, i, r correspond to $P(0)$, $P(1)$, $P(2)$ respectively. You can assume that the values for n ($P(0)$) and r ($P(2)$) are already initialized.

```
FOR $i \leftarrow 1$ to $n$
$r \leftarrow r \cdot i$
ENDFOR
```

Solution:

P	instruction	comment
1	`LOADI` 1	
2	`STORE` 1	
3	`LOAD` 0	
4	`SUBI` 1	
5	`STORE` 0	
6	`JUMP`$_=$ <19	**for** $i = 1$ to n **do**
7	`LOADI` 0	
8	`STORE` 4	
9	`LOAD` 1	
10	`STORE` 3	
11	`LOAD` 3	
12	`SUBI` 1	
13	`STORE` 3	
14	`JUMP`$_=$ <5	
15	`LOAD` 4	
16	`ADD` 2	
17	`STORE` 4	
18	`JUMP` -7	
19	`LOAD` 4	
20	`STORE` 2	$r \leftarrow r \cdot i$ $(i \geq 0)$
21	`LOAD` 1	
22	`ADDI` 1	
23	`STORE` 1	
24	`JUMP` -21	$i \leftarrow i + 1$
35	`STOP` 0	

Problem 5.2 (The Ackermann function) 8pt 8min

Write down a static $\mathcal{L}(\texttt{VM})$ procedure that computes the famous Ackermann function:

$$ack(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ ack(m-1,1) & \text{if } m > 0; n = 0 \\ ack(m-1, ack(m, n-1)) & \text{if } m > 0; n > 0 \end{cases}$$

You can assume that input is correct, i.e. $m, n \geq 0$. **Uncommented code will not be graded.**

Solution:

`proc 2 49`	$ack(m,n)$
`con 0 arg 1 leq cjp 8`	if $m \leq 0$ $(m = 0)$
`con 1 arg 2 add return`	... return $n+1$
`con 0 arg 2 leq cjp 11`	if $n \leq 0$ $(n = 0)$
`con 1 con 1 arg 1 sub call 0 return`	... return $ack(m-1,1)$
`con 1 arg 2 sub arg 1 call 0`	push $ack(m, n-1)$
`con 1 arg 1 sub call 0 return`	return $ack(m-1, ack(m, n-1))$

Problem 5.3 (TM and TCN numbers) 6pt 0min

Given a tape with an n-bit binary number written after symbol $+$ or $-$ (denoting if the number is positive or negative), design a Turing Machine which will convert it to a TCN. Initially, the head is over the sign symbol. There is no restriction where would the head be after halting. If the number of states exceeds 4, you will lose 2 points per extra state. **Uncommented code will not be graded.**

For example we would have

```
Input: -101
Output: 1011
```

Solution:

Old	Read	Wr.	Mv.	New
q_1	$+$	0	_	H
q_1	-	1	R	q_{flip}
q_{flip}	0	1	R	q_{flip}
q_{flip}	1	0	R	q_{flip}
q_{flip}	_	_	L	q_{addone}
q_{addone}	0	1	_	H
q_{addone}	1	1	L	q_{addone}
q_{addone}	_	1	_	H

---

6 Problem Solving and Search

Problem 6.1 (Missionaries and cannibals) 20pt 20min

Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. The final goal is to get everyone to the other side, without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place.

1. Formulate the problem precisely. When defining the operators, it is not necessary that you write every possible *state* → *state* combination, but you should make it clear how one would derive the next state from the current one.

2. Suppose the next-function for depth first search (DFS) and breadth first search (BFS) expands a state to its successor states using the operators you have defined in 1. in the order you have defined them. Operators that leave more cannibals than missionaries on one side will not be considered. Likewise, operators that lead to the immediate previous state will not be considered (e.g., after moving a cannibal from left to right, the next-function for this state will not include a state where a cannibal moves from right to left). Draw the search tree till depth 3. What are the first 5 nodes explored by DFS? What are the first 5 nodes explored by BFS?

3. If you would implement this problem, would you rather use BFS or DFS to find the solution? Briefly explain why?

Solution:

1. Here is one possible representation:

   The State Space is a six-tuple of integers listing the number of missionaries, cannibals, and boats on the first side, and then the second side of the river. The goal is a state with 3 missionaries and 3 cannibals on the second side. The cost function is one per action, and the successors of a state are all the states that move 1 or 2 people and 1 boat from one side to another.

   The **Initial State** is $(3, 3, 1, 0, 0, 0)$

   The **Final State** is $(\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{3}, \mathbf{3}, \mathbf{1})$

   Operators: Unless the next state leaves more cannibals then missionaries on one side, and unless the transition is impossible (eg: a movement always occurs from the side where the boat is to the other) rules are as follows:

   (a) Move a missionary to the other side.

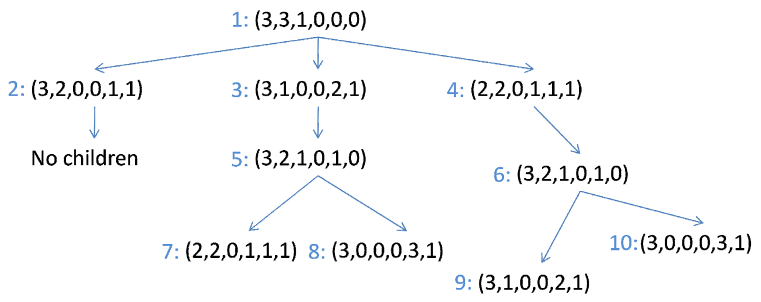
   (b) Move a cannibal to the other side.

   (c) Move 2 missionaries to the other side.

   (d) Move 2 cannibals to the other side.

   (e) Move a missionary and a cannibal to the other side.

2. Depends in which order the operators are defined. In the case above, it will look as follows:



3. BFS, since the branching factor is not big (we won't have memory problems), and DFS may get stuck in loops.

Problem 6.2 (Relaxed Problem) 8pt 8min

The relaxed version of a search problem P is a problem P' with the same states as P, such that any solution of P is also a solution of P'. More precisely, if s' is a successor of s in P, it is also a successor in P' with the same cost. Prove or refute that for any state s, the cost $c'(s)$ of the optimal path between s and the goal in P' is an admissible A^* heuristic for P.

Hint: Think about the graphical representation of the problems.

Solution: Graphically, P' has all the arcs from P, plus maybe more. This means that the optimal path in P' is the same as the optimal path in P, or better through the possibility of using the additional arcs. Therefore $c'(s) \leq c(s)$, which is the admissibility criterion for a heuristic.

Problem 6.3 (Global Solutions) 6pt 6min

For each of the following algorithms, briefly state why or why not they are guaranteed to converge to a global optimum on a problem P:

1. A^* search with the heuristic from the problem above
2. Greedy search with the same heuristic
3. Hill Climbing
4. Genetic Algorithms

Solution:

1. search with the heuristic from the problem above: will converge to a global solution because the heuristic is admissible
2. Greedy search with the same heuristic: no, it actually might not converge at all because it can get stuck in infinite loops, as it doesn't take into account the distance so far, like A^*
3. Hill Climbing: no, it will just climb the current "hill", which does not necessarily have the biggest "hight"
4. Genetic Algorithms: no, there is no guarantee that GA's find the optimal solution. The quality of a solution of course depends on how cross-overs, mutations and other settings are chosen, and it can happen that the population simply cannot reach the region in the solution space in which the global optimum is.

---

7 Prolog

Problem 7.1 (Prime numbers) 8pt 8min

Write a ProLog predicate `myPrime(X)` that is true only if X is prime. Consider only positive integer inputs.

Note:

- 1 is not a prime number.
- The built-in ProLog operator `mod` may be useful: `X is A mod B`.

Solution:

```
has_divisors(1).
has_divisors(A) :- B is A-1, between(2,B,D), (0 is A mod D).
myPrime(X) :- not(has_divisors(X)).

% Here is an alternative solution by Ankur Modi
isprime(X,X).
isprime(X,N):- X>N ,Y is X mod N, not(Y=0),N2 is N+1,isprime(X,N2).
myPrime(X):-isprime(X,2).
```