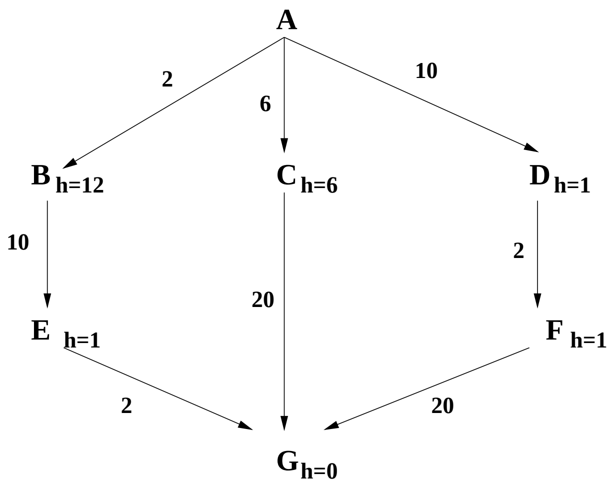
Key for 1st Midterm Exam

1. *15 points*

   You are given the following graph, where each node has an identifier (a letter) and an h value. A number along an arc indicates the cost of the arc.



   (a) Show in what order A* expands nodes from Start to G. G is the goal node. For each node expanded during the search show its f and g values. If a node is reached on multiple paths show its f and g values each time the node is reached, and indicate its parent node.

   Answer:

step	node expanded	g	h	f	parent	open	comments
1	A	0	?	?		[]	
2	B	2	12	14	A		
	C	6	6	12	A		
	D	10	1	11	A	D C B	
3	F	12	1	13	D	C F B	
4	G	26	0	26	C	F B G	
5	G	32	0	32	F	B G	discarded since previous G has lower cost
5	E	12	1	13	B	E G	
6	G	14	0	14	E	G	previous G discarded since it has higher cost
7	G is goal						

   (b) What is the solution path found?

   Answer: The solution is A-B-E-G.

   (c) Is the h function admissible? is it consistent? Justify your answer.

   Answer: Yes, h is admissible. No, h is not consistent. To see why we can look for decreasing f values on any path. The triangle inequality is violated for nodes B and E.

2. *15 points*
   Suppose you decide to do best-first search using the following evaluation function $f(n) = (1-w)g(n) + wh(n)$.

   (a) Assuming that $h(n)$ is admissible, what are the values of w that guarantee the algorithm will find an optimal solution? Justify your answer.

   Answer:

   - with $w = 0$ the algorithm is uniform cost, which is guaranteed to find an optimal solution if there is a solution.
   - with $w = 0.5$ the algorithm is A*, which, given that h is admissible, is guaranteed to find an optimal solution if there is a solution.

   (b) Is there a range of values of w which guarantees that the algorithm using the evaluation function f is admissible? If yes, what is the range? Justify your answer.

   Answer: Assuming that h is admissible, with values of w in the range 0 to 0.5 the algorithm is guaranteed to find an optimal solution (if there is a solution). To see why, we can observe that for $0 \leq w \leq 0.5$ g a coefficient larger than the coefficient for h. So, assuming h is admissible, the result is the same as if we had scaled down the f function. This is no longer true when $w > 0.5$, since we are multiplying h by a factor larger than the factor used for g.

3. *15 points*
   Suppose you have two admissible heuristics, h_1 and h_2. You decide to create the following new heuristic functions defined as follows:

$$h_3(n) = max(h_1(n), h_2(n))$$

$$h_4(n) = max(h_1(n), 1.1 \times h_2(n))$$

$$h_5(n) = min(h_1(n), 3 \times h_2(n))$$

$$h_6(n) = \frac{h_1(n) + h_2(n)}{2}$$

   For each of the new heuristics specify of it is admissible or not. Justify your answer. Would you use any of these heuristics instead of using h_1 or h_2?

   Answer:

   - $h_3(n) = max(h_1(n), h_2(n))$ is admissible, since given that $h_1(n) \leq h^*(n)$ and $h_2(n) \leq h^*(n)$ we deduce $max(h_1(n), h_2(n)) \leq h^*(n)$
   - $h_4(n) = max(h_1(n), 1.1 \times h_2(n))$ is not admissible, since we cannot say that $1.1 \times h_2(n) \leq h^*(n)$
   - $h_5(n) = min(h_1(n), 3 \times h_2(n))$ is admissible, since given that $h_1(n) \leq h^*(n)$ it is always true that $min(h_1(n), \infty) \leq h^*(n)$
   - $h_6(n) = \frac{h_1(n)+h_2(n)}{2}$ is admissible, since given that $h_1(n) \leq h^*(n)$ and $h_2(n) \leq h^*(n)$ we deduce $\frac{h_1(n)+h_2(n)}{2} \leq h^*(n)$

4. *15 points*
   Answer these questions briefly but precisely.

(a) Would Hill-Climbing be appropriate for the Missionaries and Cannibals Problem? Why (or why not?).

Answer: No, it is not appropriate. To solve the problem some people have to come back across the river, which means that to reach a solution the search algorithm has to explore states that are not as good as the current one.

(b) Is it possible for Iterative Deepening Depth-First Search to do worse than Depth-First? Explain your reasoning.

Answer: Yes, when the branching factor is small. Think about the case when the branching factor is exactly 1. Iterative Deepening will expand multiple times the states from start, while Depth-First will just go down the path once.

(c) Explain briefly when you would use LRTA* instead of Online-DFS.

Answer: LRTA* encourages exploration of apparently promising paths, and so it should be used when exploration is useful (e.g. the space is large and there are many solution paths) and when a reasonable heuristics exists.

5. *10 points*

Write a function, `remove-adj-dup`, to remove all adjacent duplicate elements in a list. It should work like this:

```
(remove-adj-dup '(a b b c b d d d) )
(a b c b d)
```

Answer:

```
(defun remove-adj-dup (l)
  (cond ((null l) nil)
        ((null (cdr l)) l)
        ((eql (car l) (cadr l)) (remove-adj-dup (cdr l)))
        (t (cons (car l) (remove-adj-dup (cdr l))))))
```

6. *5 points*

Write a function, `add--numbers`, to add all the values in an association list that are numbers. It should work like this:

```
(add-all-numbers '((color red) (weight 3) (type apple) (id  120452)))
120455
```

Answer:

```
(defun add-all-numbers (l)
  (reduce #'+
          (mapcar #'(lambda (x)
                            (if (numberp (cadr x))
                                (cadr x)
                                 0))
                  l)))
```