6.034 Quiz 1
25 September 2013

Name	JADE HARLEY
email	

Circle your TA **(for 1 extra credit point)**, so that we can more easily enter your score in our records and return your quiz to you promptly.

Michael Fleder	**Giuliano Giacaglia**	**Dylan Holmes**
Casey McNamara	**Robert McIntyre**	**Duncan Townsend**
Mark Seifter	**Sam Sinai**	**Prashan Wanigasekara**

Problem number	Maximum	Score	Grader
1	50		
2	50		
Total	100		

There are 11 pages in this quiz, including this one, but not including blank pages and tear-off sheets. Tear-off sheets with duplicate drawings and data are located after the final page of the quiz. As always, open book, open notes, open just about everything, including a calculator, but no computers.

Problem 1: Rule systems (50 points)

After releasing his thirteenth studio album, eminent yet conscientious musician Jay-Z begins to wonder if he has achieved success in life yet, or if he has more work to do. Having acquired many skills in 6.034, you decide to help him out by writing a rule-based system for determining success.

Rules:

```
P0    IF    (OR ‘(?x) is wealthy’
                ‘(?x) is a very important political figure')
      THEN ‘(?x) is powerful’

P1    IF     (AND ‘(?x) is married to (?y)’,
                  ‘(?y) is (?x)'s soul mate’)
      THEN  ‘(?x) has a happy marriage’

P2    IF     (AND ‘(?x) has a beautiful daughter’,
                 ‘(?x) has a happy marriage’)
      THEN   ‘(?x) has a happy home’

P3    IF    ‘(?x) is (?y)'s soul mate’
      THEN ‘(?y) is (?x)'s soul mate’

P4    IF    ‘(?x) is not famous’
      THEN ‘(?x) is safe’

P5    IF    (AND ‘(?x) is powerful’
                 ‘(?x) has a happy home’
                 ‘(?x) is safe’)
      THEN ‘(?x) is successful’
```

Assertions:

A0: Jay-Z is wealthy
A1: Jay-Z is Beyoncé's soul mate
A2: Jay-Z is married to Beyoncé
A3: Jay-Z has a beautiful daughter

For your convenience, a copy of these rules is provided on a tear-off sheet after the end of the exam.

Part A: Backward chaining (25 points)

Make the following assumptions about backward chaining:

- The backward chainer tries to find a matching assertion in the list of assertions. If no matching assertion is found, the backward chainer tries to find a rule with a matching consequent. In case none are found, then the backward chainer assumes the hypothesis is false.
- The backward chainer never alters the list of assertions, so it can derive the same result multiple times.
- Rules are tried in the order they appear.
- Antecedents are tried in the order they appear.
- Lazy evaluation/short circuiting is in effect.

First, determine whether Jay-Z is successful by simulating backward chaining with the hypothesis

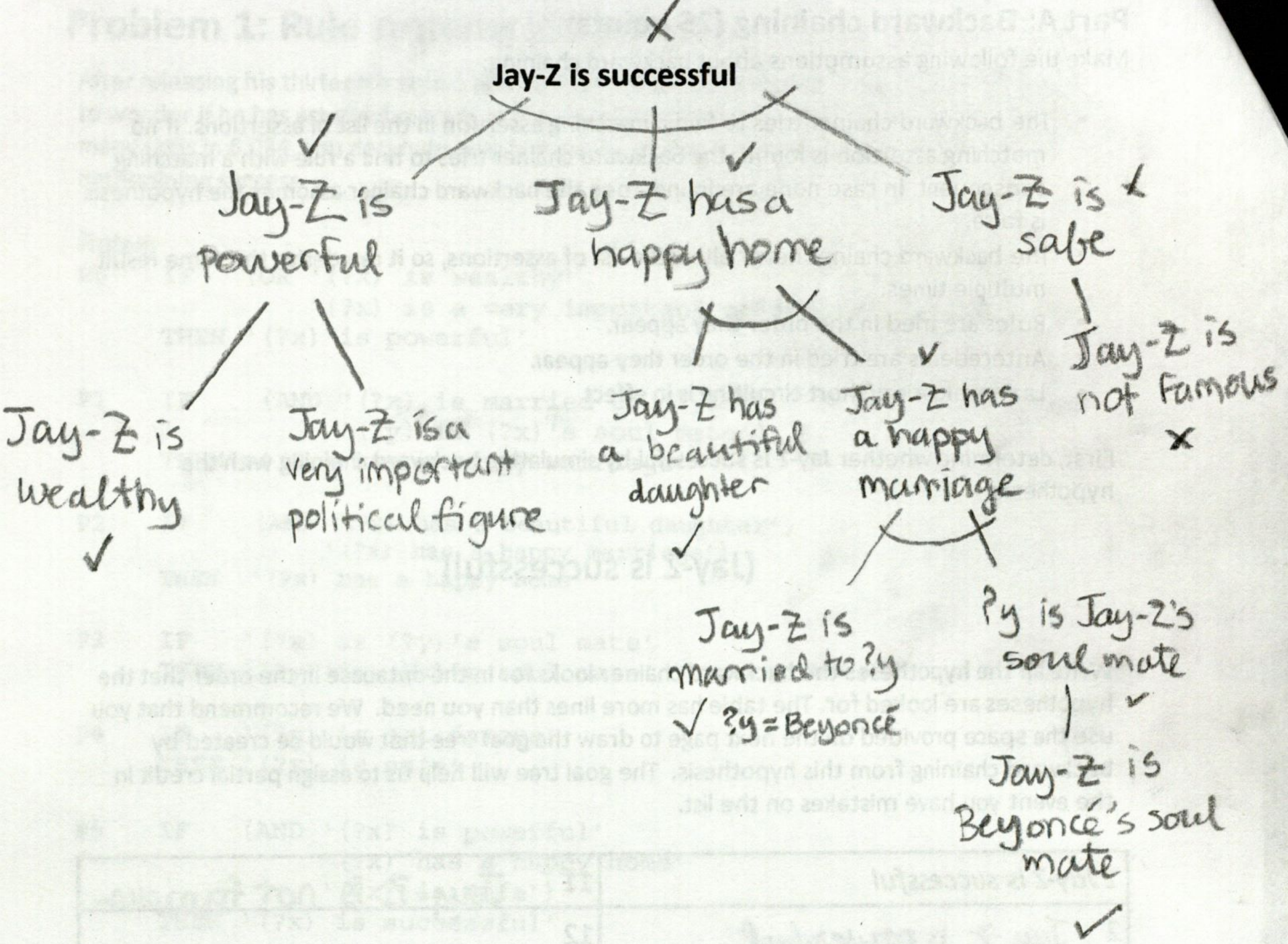
(Jay-Z is successful)

Write all the hypotheses the backward chainer looks for in the database in the order that the hypotheses are looked for. The table has more lines than you need. We recommend that you use the space provided on the next page to draw the goal tree that would be created by backward chaining from this hypothesis. The goal tree will help us to assign partial credit in the event you have mistakes on the list.

1 *Jay-Z is successful*	11 Jay-Z is not famous
2 Jay-Z is powerful	12
3 Jay-Z is wealthy	13
4 Jay-Z has a happy home	14
5 Jay-Z has a beautiful daughter	15
6 Jay-Z has a happy marriage	16
7 Jay-Z is married to ?y	17
8 Beyoncé is Jay-Z's soulmate	18
9 Jay-Z is Beyoncé's soul mate	19
10 Jay-Z is sage	20

TA note: In #7, the backward chainer looks for "Jay-Z is married to ?y". It has not yet bound the var ?y = Beyoncé which it will do after this step, having found the assertion "Jay-Z is married to Beyoncé."

Space provided to draw the goal tree.



Part B: Forward chaining (25 points)

Make the following assumptions about forward chaining:

- Assume rule-ordering determines which rule fires when multiple rules trigger
- New assertions are added to the bottom of the list of assertions.
- If a particular rule matches assertions in the list of assertions in more than one way, the matches are considered in the order corresponding to the top-to-bottom order of the matched assertions. Thus, if a particular rule has an antecedent that matches both A1 and A2, the match with A1 is considered first.

Run forward chaining on the rules and assertions on page 2.

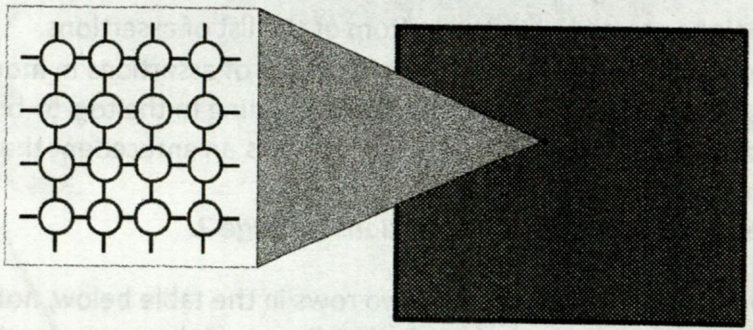
For the first two iterations, fill out the first two rows in the table below, noting the rules whose antecedents match the data, the rule that fires, and the new assertions that are added by the rule. For the remainder, supply only the fired rules and new assertions. There are more rows in the table than you'll need.

	Matched	Fired	New Assertions Added to the List of Assertions
1	P_0 P_3	P_0	Jay-Z is powerful
2	P_0 P_3	P_3	Beyoncé is Jay-Z's soulmate
3		P_1	Jay-Z has a happy marriage
4		P_2	Jay-Z has a happy home
5			
6			
7			
8			
9			

Problem 2: Optimal Search (50 points)

Part A: The big picture (25 points)

Suppose you are performing search on a very densely-packed grid, a sample of which is shown below:



You are going to visually compare the results of several search algorithms using the following coloring scheme: Before you run each algorithm, you'll paint all the nodes in the graph white. Every time you extend a node, you'll paint it gray. (If you extend a node more than once, it will remain gray.) This will generate a picture for each search algorithm.

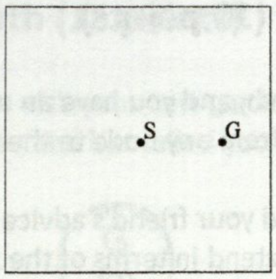
Below is a list of algorithms, and on the next page is a collection of pictures. Match each search algorithm in the table with the picture it makes when finding the goal. There is only one best answer for each algorithm, but some of the pictures may be used more than once, and some of the pictures may not be used at all.

Assumptions: All edges have length one. **Break ties by trying up, right, left, and down children in that order.** **Importantly**, you are to use ***half* the Manhattan distance to the goal** as your heuristic estimate—recall that the Manhattan distance between two points is defined by

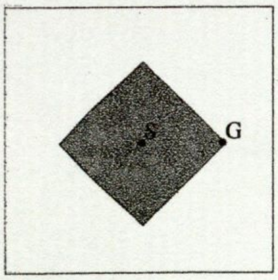
$$d(p,q) = |p_x - q_x| + |p_y - q_y|.$$

Search algorithm	**Best matching picture** (Circle one)
Depth-first search (with backtracking, no extended set)	A B [✓ C] D E F
Breadth-first search (with no extended set)	[✓ A] B C D E F
Hill climbing (with backtracking, no extended set)	A B C [✓ D] E F
Branch and bound with an extended set, but no heuristic	[✓ A] B C D E F
A* search	A B C D E [✓ F]

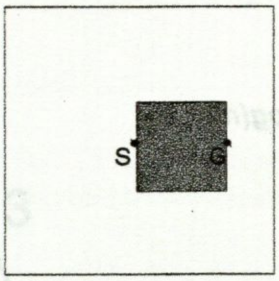
Note: For your convenience, a tear-off sheet with several copies of a grid graph are provided after the last page of the exam. You are not required to use them, and you do not need to turn them in.



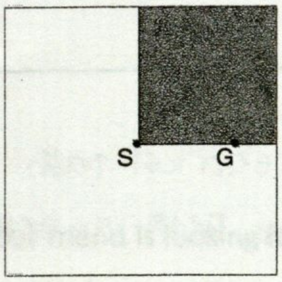
Initial state: before search, the grid looks like this.



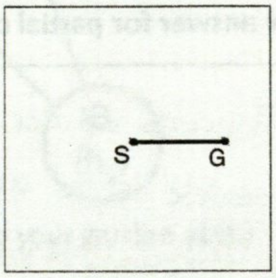
Picture A
(a diamond)



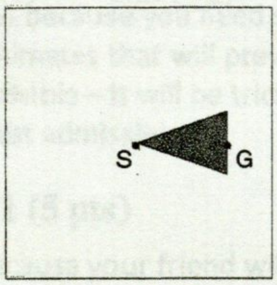
Picture B
(a square)



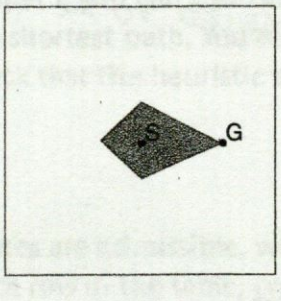
Picture C
(a corner)



Picture D
(a horizontal line segment)



Picture E
(a triangular fan)



Picture F
(a kite)

Part B: Dead-on Reckoning (10 points)

Suppose you are performing A* search, and you have an all-knowing friend who will tell you the true length of the shortest path from any node to the goal.

When performing search with A* and your friend's advice as your heuristic estimate, about how many nodes do you expect to extend in terms of the number of nodes (*n*) in the optimal path from the start node to the goal node? Assume that there are never any "ties" when evaluating which node to extend next.

[✓ **(A)**] About *n*

(B) About *n log(n)*

(C) About n^2

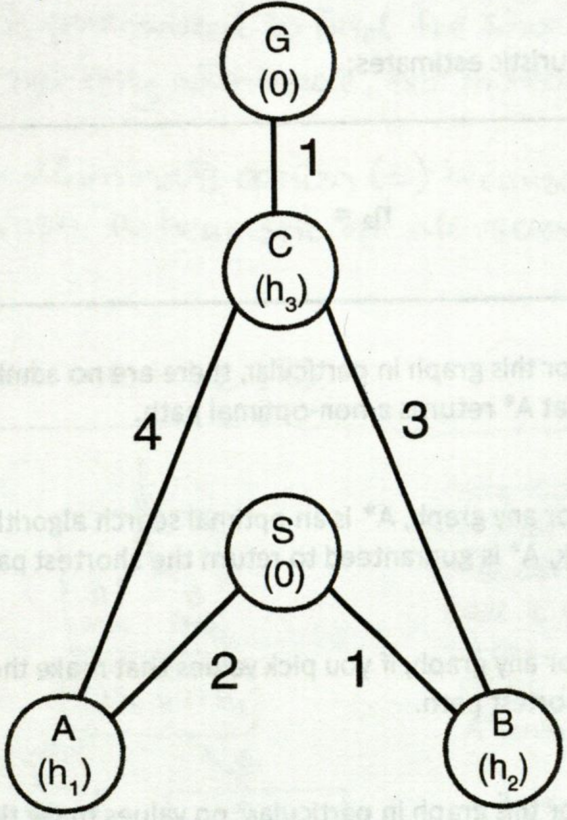
(D) It depends on the search space

Briefly explain your answer for partial credit:

With my friend's advice, I know that my estimate at each node is the actual length of the shortest path to the goal through that node. I choose the best of these always (and there are no ties), so I thereby get the shortest path to the goal with no wrong turns.

Part C: Down the garden path (15 points)

A week ago, you invited your friend to visit your new garden—but alas, you forgot all about the visit, and now that friend is on her way and has called you to ask for directions. You begin by describing the following map:



(Your friend is looking for a path from her house at **S**, to your garden at **G**.)

Knowing that **your friend will use A* search** (breaking ties lexicographically), you plan to provide her with heuristic estimates (h_1, h_2, h_3) of the remaining distance to **G** at each node. But because you need more time to get ready, you wonder if you can choose heuristic estimates that will prevent your friend from finding the shortest path. You're not sure it's possible—it will be tricky because she will certainly check that the heuristic estimates are at least admissible.

C1 (5 pts)

Because your friend will check that the heuristic estimates are admissible, which of the following values, if any, should you avoid using? For each row in the table, **cross out any values that are inadmissible. If all of them are admissible, circle "All of these are admissible" instead.**

h_1	0	1	2	3	4	5	~~6~~	~~7~~	All of these are admissible
h_2	0	1	2	3	4	~~5~~	~~6~~	~~7~~	All of these are admissible
h_3	0	1	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~	~~7~~	All of these are admissible

C2 (10 pts)

Find non-negative, integer values for the heuristic estimates h_1, h_2, and h_3 such that **the heuristic is admissible, but A* search returns a path that isn't the shortest path.** Circle the best response from the answers below. If you circle answer (A), also write down the values of the heuristic estimates you found.

(A) Use the following heuristic estimates:

h_1 =	h_2 =	h_3 =

[✓ **(B)**] It's impossible—for this graph in particular, there are no admissible heuristic estimates such that A* returns a non-optimal path.

(C) It's impossible—for any graph, A* is an optimal search algorithm; no matter which estimates you pick, A* is guaranteed to return the shortest path.

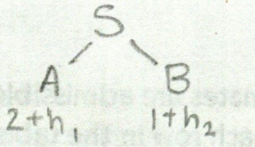
(D) It's impossible—for any graph, if you pick values that make the heuristic admissible, A* will find the shortest path.

(E) It's impossible—for this graph in particular, no values make the heuristic admissable.

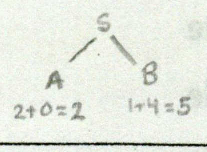
Show your work for partial credit. (there's additional space on the next page)

Your goal is to make A* search produce path SACG instead of the shortest path SBCG.

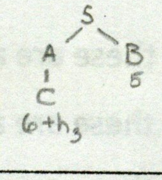
Initially, your search tree looks like:



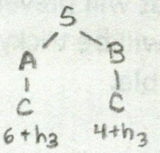
You can pick h_1 to be as small as possible (= 0) and h_2 as large as possible (=4) to ensure that A is extended first.



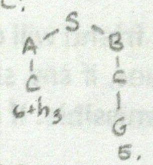
⇒



Now, because h_3 must be non-negative, B will be extended next no matter what.



Then SBC:



Then SBCG.

Despite your best efforts, A* found the shortest path.

==>

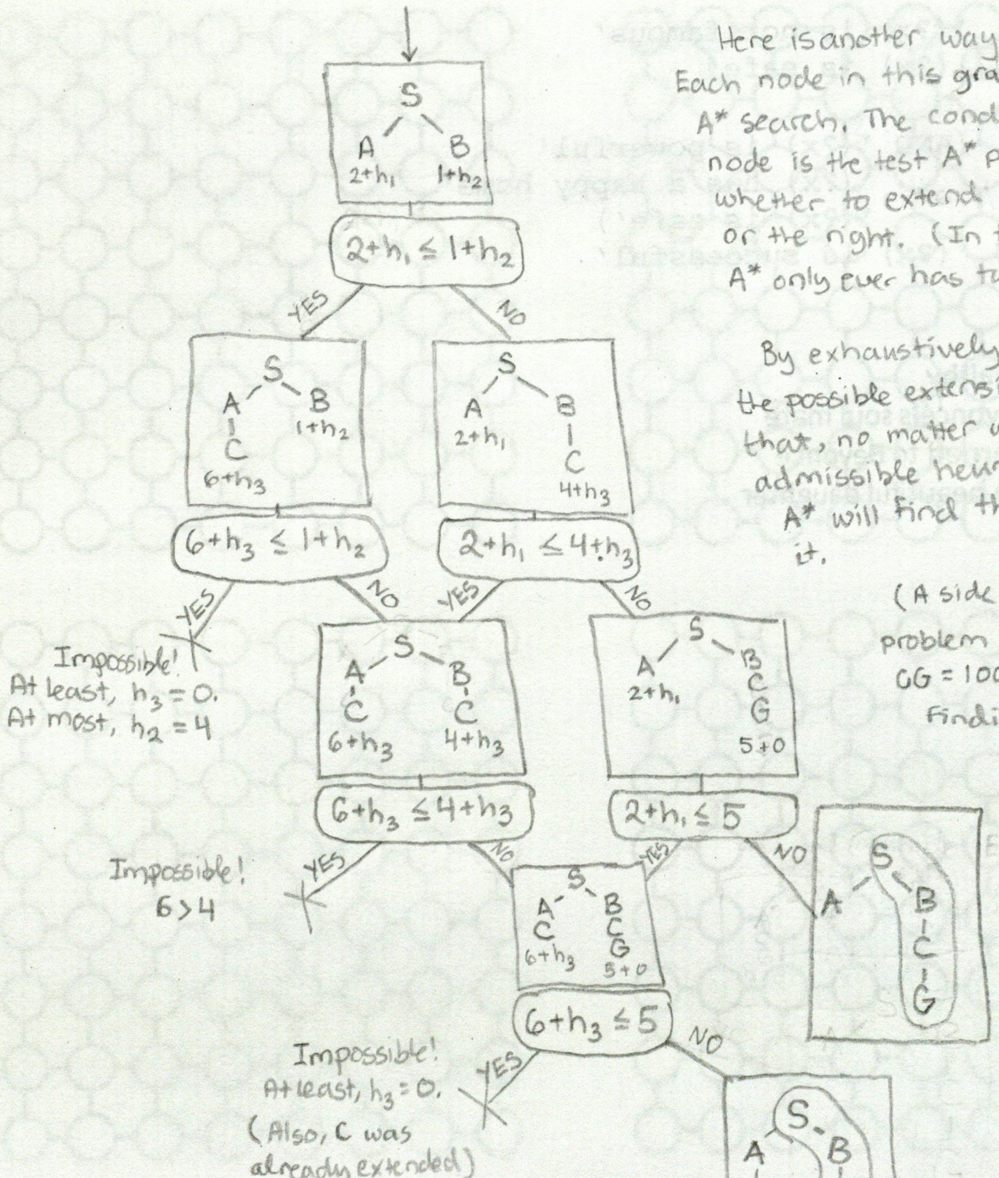
Space provided below so you can show your work on problem C2 if you wish.

So, we can eliminate option (A).

We can also eliminate options (C) and (D) because we know that A* is guaranteed to find the shortest path only if the heuristic is not only admissible, but moreover consistent.

Finally, we can eliminate option (E) because for this graph and for any graph, a heuristic of all zeroes is always admissible.

Thus, the answer must be (B).



Here is another way to look at it. Each node in this graph is a state of A* search. The condition under each node is the test A* performs to decide whether to extend the left branch or the right. (In this simple problem, A* only ever has two choices.)

By exhaustively considering all the possible extensions, we can see that, no matter what, if you pick admissible heuristics for this graph, A* will find the shortest path through it.

(A side note: if you change the problem to make the length of CG = 1000, you can trick A* into finding a suboptimal path even if the heuristic is admissible.)