6.034 Quiz 1
26 September 2014

Name	
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.

Josh Blum	**Elisa Castañer**	**Pedro Cattori**
Malcom Gilbert	**Dylan Holmes**	**Jessica Noss**
Theophilus Teeyay	**Duncan Townsend**	**Siyao Xu**

Problem number	Maximum	Score	Grader
1	40		
2	60		
Total	100		

There are 12 pages in this quiz, including this one, but not including 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.

This page contains no quiz material.

Problem 1: Rule-based systems (40 points)

Because roommates are often unpredictable, Josh decides to use his knowledge of rule-based systems to help him predict their behavior. Based on his roommates' previous actions, he constructs a list of rules and a list of assertions about them, as follows:

Rules:

```
P0   IF    OR(‘(?A) plays basketball',
              ‘(?A) is lazy’),
     THEN  ‘(?A) hasn't eaten’

P1   IF    ‘(?A) likes to code’,
     THEN  ‘(?A) is working’

P2   IF   AND( ‘(?A) isn't busy’,
               ‘(?B) enjoys playing basketball’),
     THEN  ‘(?A) plays basketball’

P3   IF   AND( ‘(?A) isn't busy’,
               NOT(‘(?A) is lazy’)),
     THEN ‘(?A) will get his ID’

P4   IF   AND( ‘(?A) hasn't eaten’,
               ‘(?A) isn't busy’,
               ‘(?B) likes to cook’
                NOT(‘(?A) plays basketball’)),
     THEN ‘(?A) has dinner with (?B)’
```

Assertions for Part A (backward chaining):

A0: Josh is lazy
A1: Mother Aryan enjoys playing basketball
A2: William likes to cook
A3: Louis likes to code

For your convenience, a copy of these rules and assertions is provided on a tear-off sheet after the last page of the quiz.

Part A: Backward chaining (17 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.

A1 (14 points)

Simulate backward chaining with the hypothesis:

Josh has dinner with William

Write all the hypotheses the backward chainer looks for in the database in the order that the hypotheses are looked for. The table may have 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 *Josh has dinner with William*	9
2	10
3	11
4	12
5	13
6	14
7	15
8	16

A2 (3 points)

Does Josh have dinner with William? (Does backward chaining prove the hypothesis “Josh has dinner with William”?) Circle one:

YES NO

Space provided to draw the goal tree.

Josh has dinner with William

Part B: Forward chaining (23 points)

Make the following assumptions about forward chaining:

- Assume rule-ordering conflict resolution
- 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.

B1 (20 points)

Patrick, Josh's new roommate, has just moved into town. Josh quickly adds a new assertion to the list to account for his first impression of this unknown character.

Assertions for Part B (forward chaining):

A0: Josh is lazy
A1: Mother Aryan enjoys playing basketball
A2: William likes to cook
A3: Louis likes to code
A4: Patrick isn't busy

Perform forward chaining using the rules mentioned on page 3 and **<u>this new list</u>** of assertions. 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 may be more rows in the table than you'll need.

	Matched	Fired	New Assertions Added to the List of Assertions
1			
2			
3	▨		
4	▨		
5	▨		
6	▨		
7	▨		
8	▨		

B2 (3 points)

Given the assertions in part B1, which of the rules (P0, P1, P2, P3, P4) will never fire during forward chaining? (If all of the rules will fire, write NONE instead.)

Problem 2: Optimal Search (60 points)

Part A: Search me (9 points)

A1. You have a summer internship with a travel agency. Your first task is to find an automobile route from Boston to San Francisco, but not just any route: you have to find the *most scenic* route.

- The road connecting any two cities has a scenic value, so the most scenic route is the one whose sum of city-to-city scenic values is **maximum**.
- There is no limit to how scenic a road connecting two cities can be.
- Your customers want to feel like they are making progress, so your route must include only connections traveled from east to west.
- You are to work with a map that includes only major cities and the roads connecting them, so you are not worried about any exponential blowup that could make certain search methods computationally infeasible, but of course you would like your search method to be as efficient as possible.

Of the following, your best choice is (circle the **single best** answer):

1 British Museum algorithm
2 Hill climbing
3 Bi-directional breadth-first search.
4 Bi-directional depth-first search.
5 Branch and bound search

Note: a *bi-directional search* expands one path starting from S, then one from G, then one from S, and so on, until a terminal node on a path from S is the same as a terminal node on a path from G.

A2. Your second task is to find an automobile route from point S to point G. This time, you want to find the route that goes through the fewest cities. Of course, you would like to be as efficient as possible.

Of the following, your best choice is (circle the **single best** answer):

1 British Museum algorithm
2 Hill climbing
3 Bi-directional breadth-first search.
4 Bi-directional depth-first search.
5 Branch and bound search

A3. Completely exhausted after your first, hard day at the travel company, which is in Paris, you have to walk to your apartment because the taxis are on a one-day strike. You are not sure how to get there, and you have no map, but your apartment is near the Eiffel tower, visible everywhere in the city.

Of the following, your best choice is (circle the **single best** answer):

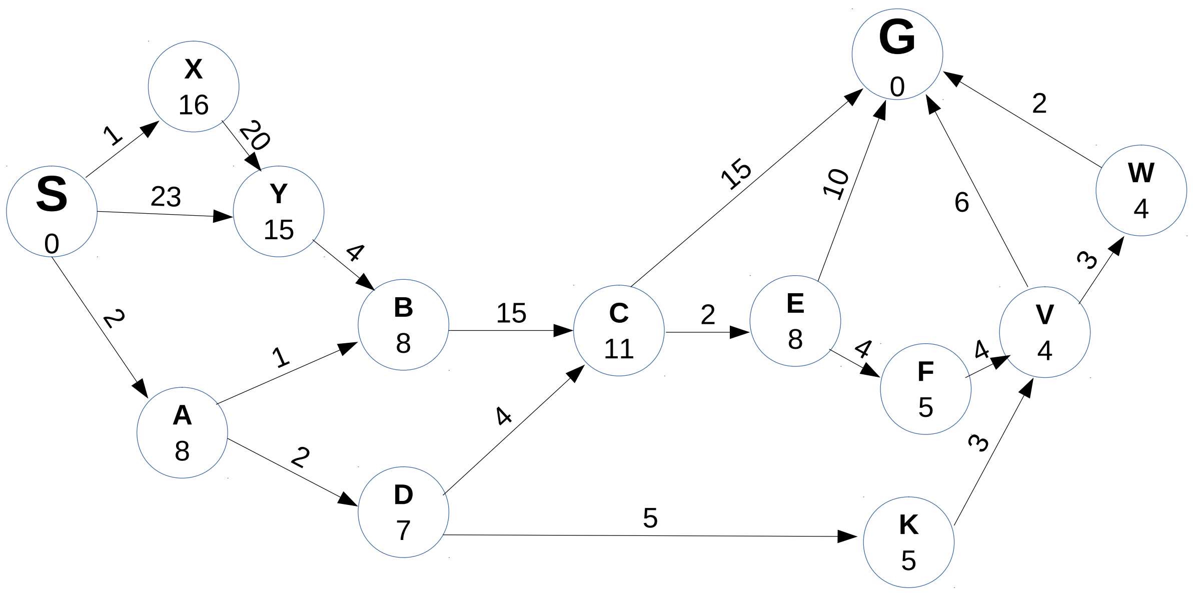
1 British Museum algorithm
2 Hill climbing
3 Bi-directional breadth-first search.
4 Bi-directional depth-first search.
5 Branch and bound search

Part B: So busted! (17 points)

B1 (14 points)

Long-suffering teen Candace has just discovered her siblings goofing off and must race home to tell their mother. She decides to use **branch and bound, with no heuristic and no extended set**, to find the quickest route home starting from node **S** and ending at node **G**.

Referring to the graph below, use branch and bound (with no heuristic and no extended set) to find a route from node S to node G. The numbers inside each node indicate heuristic estimates of distance to the goal, which you will not need in this part. For credit, **draw your search tree** on the following page.



- **Draw the children of each node in alphabetical order**, and **break ties using lexicographic/dictionary order** (for example, if two paths S-P-U and S-Q-T are tied, you should extend path S-P-U before path S-Q-T).
- **Note #1:** For your convenience, a copy of the graph is provided on a tear-off sheet after the last page of the quiz.
- **Note #2:** Notice that the edges in this graph are **one-way**; paths must only go in the direction of the arrows.

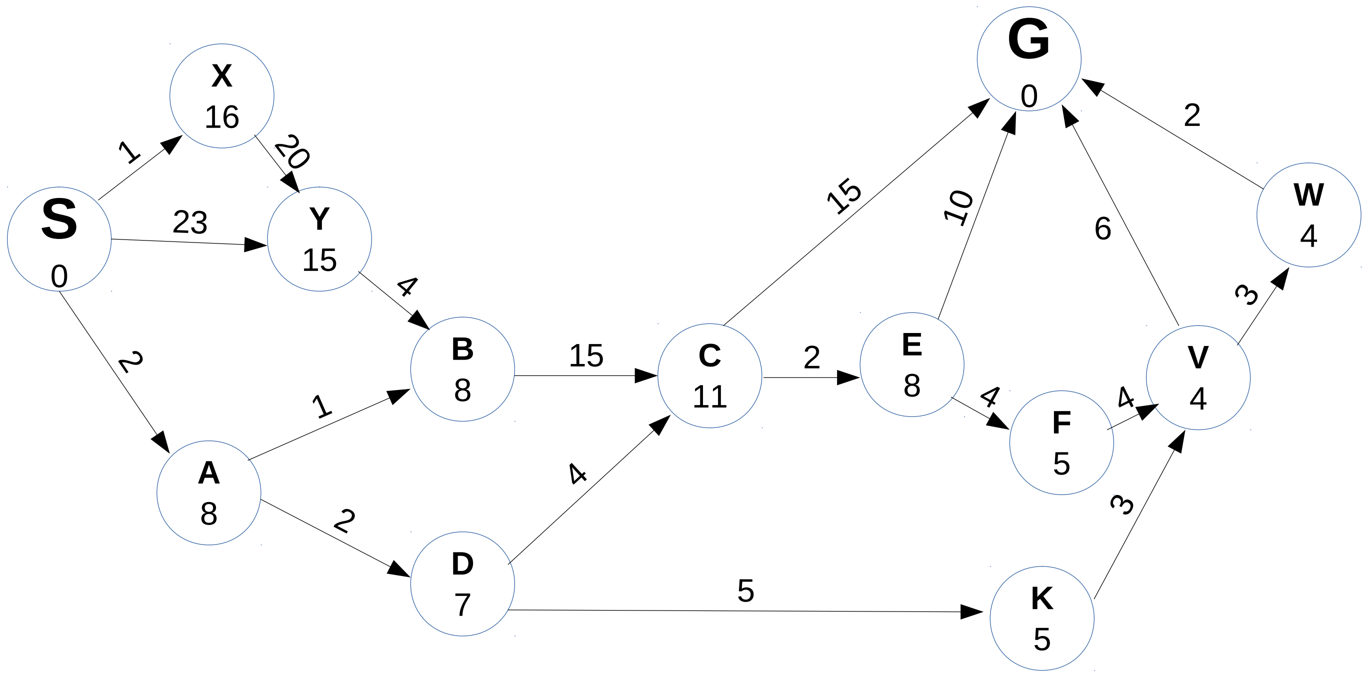
B2 (3 points)
What path does Candace find using branch and bound?

What is its total path length?

Part C: Scram! (25 points)

Not eager to get scolded by their mother, Candace's siblings decide to look for a way home themselves—so, they ask an ostrich herder for heuristic estimates of the distance home from each node, then decide to use **A* search** (that is: branch and bound, with a heuristic and with an extended set) to look for a way home. The numbers inside each node indicate heuristic estimates of distance to the goal, which you will need in this part.

(Here is a copy of the graph that was used in part B)



C1 (14 points)
Use A* search to find a route from node S to node G. For credit, **draw your search tree** below.

- **Draw the children of each node in alphabetical order**, and **break ties using lexicographic/dictionary order** (for example, if two paths S-P-U and S-Q-T are tied, you should extend path S-P-U before path S-Q-T).
- **Note #1:** For your convenience, a copy of the graph is provided on a tear-off sheet after the last page of the quiz.
- **Note #2:** Notice that the edges in this graph are **<u>one-way</u>**; paths must only go in the direction of the arrows.

C2 (6 points)
List the extended nodes in the order you extended them during A* search.

Important! The quiz continues on the next page. ↪

C3 (3 points)
What path do the siblings find using A* search?

What is its total path length?

C4 (2 points)
Regardless of your answers to parts C1 and C2, assume that branch and bound found a shorter path than A*. Very briefly, explain this result.

Tear-off sheet for Problem 1: Rule-based systems

Rules:

```
P0   IF    OR(‘(?A) plays basketball',
               ‘(?A) is lazy’),
     THEN  ‘(?A) hasn't eaten’

P1   IF    ‘(?A) likes to code’,
     THEN  ‘(?A) is working’

P2   IF   AND( ‘(?A) isn't busy’,
               ‘(?B) enjoys playing basketball’),
     THEN  ‘(?A) plays basketball’

P3   IF   AND( ‘(?A) isn't busy’,
               NOT(‘(?A) is lazy’)),
     THEN ‘(?A) will get his ID’

P4   IF   AND( ‘(?A) hasn't eaten’,
               ‘(?A) isn't busy’,
               ‘(?B) likes to cook’
                NOT(‘(?A) plays basketball’)),
     THEN ‘(?A) has dinner with (?B)’
```

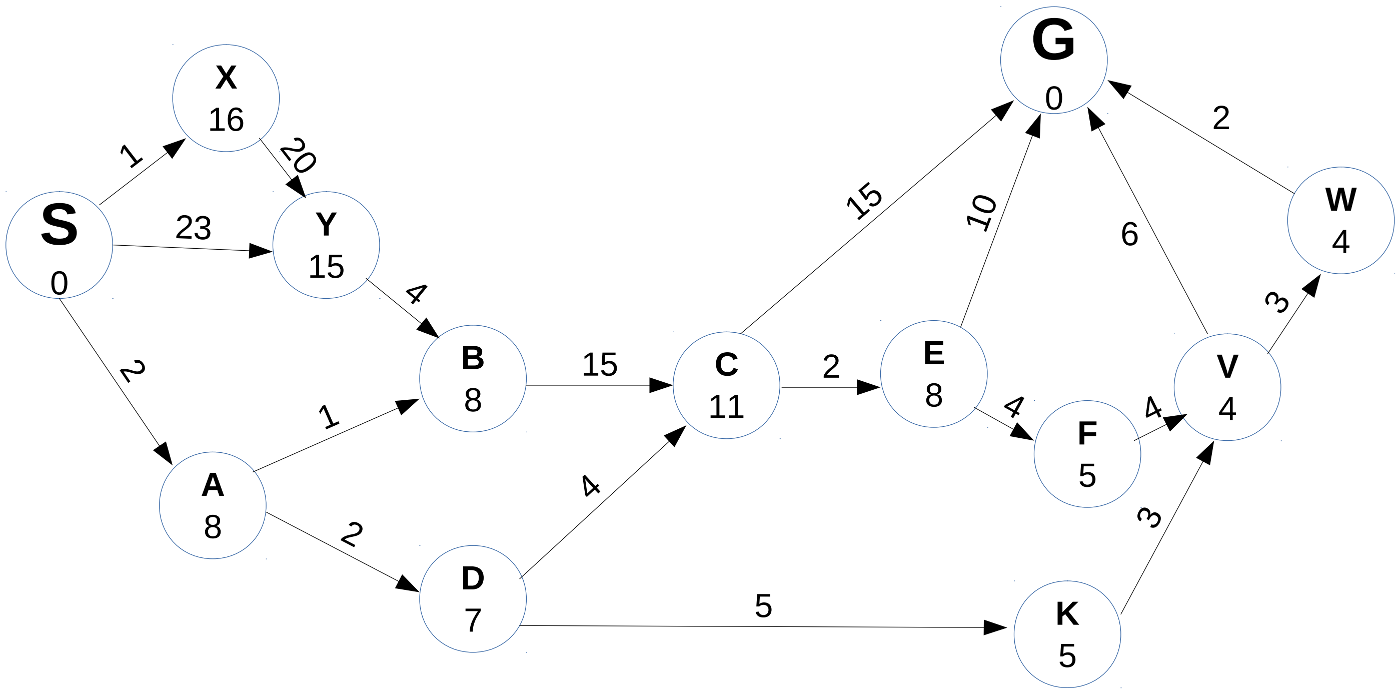
Assertions for Part A (backward chaining):

A0: Josh is lazy
A1: Mother Aryan enjoys playing basketball
A2: William likes to cook
A3: Louis likes to code

Assertions for Part B (forward chaining):

A0: Josh is lazy
A1: Mother Aryan enjoys playing basketball
A2: William likes to cook
A3: Louis likes to code
A4: Patrick isn't busy

Tear-off sheet for Problem 2: Search
(Parts B and C only)



S
0
X
16
Y
15
A
8
B
8
D
7
C
11
E
8
F
5
K
5
V
4
W
4
G
0
1
23
2
20
4
1
2
15
4
5
15
2
10
4
4
3
6
3
2