Review pt 2

1) Best first Search and A* [10]

Consider the search space below, where *S* is the start node and *G1* and *G2* satisfy the goal test. Arcs are labeled with the cost of traversing them and the estimated cost to a goal (the h function itself) is reported inside nodes.
For each of the following search strategies, indicate which goal state is reached (if any) and list, *in order*, all the states *popped off of the OPEN list*. When all else is equal, nodes should be removed from OPEN in alphabetical order.

a) Best-First-Search (using function h only) [3]

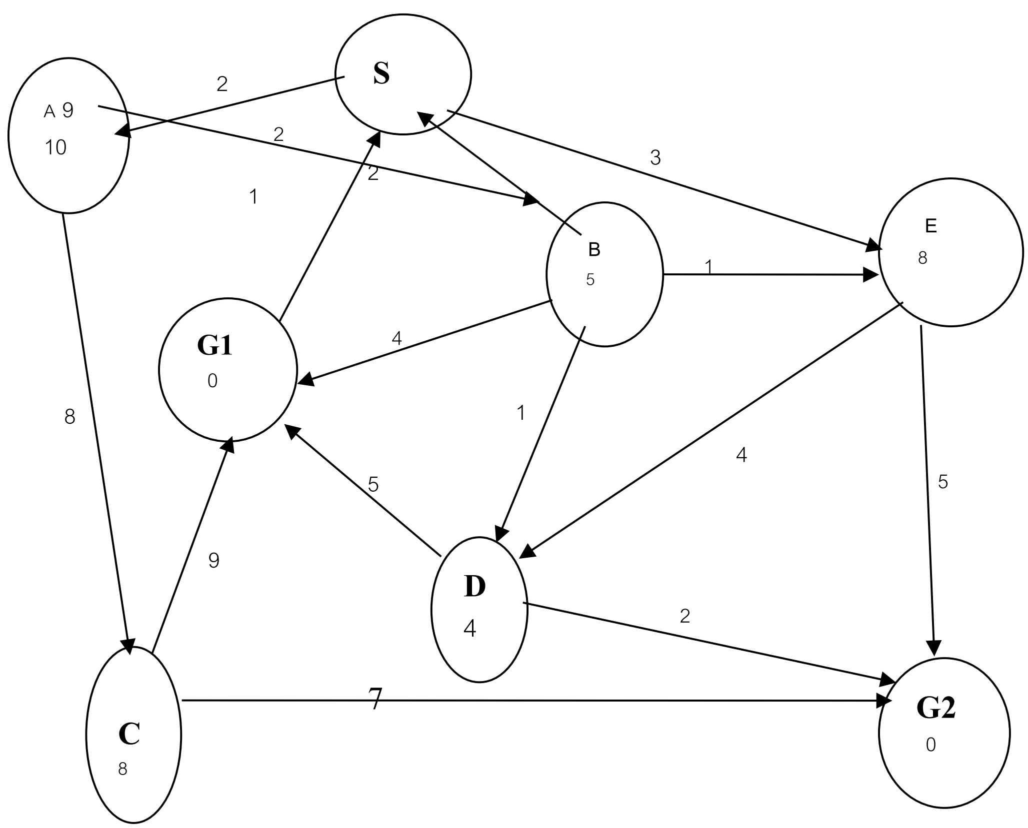
Goal state reached: G2 [1]

States popped off OPEN: S, E, G2 [2]

b) A* (using f=g+h)[4]

Goal state reached: G1 [1]

States popped off OPEN: S, A, B, G1 [3]



c) Assume you have 2 admissible heuristics h1(s) and h2(s) are given for a given seach problem. Is h3(s)=min(h1(s),h2(s)) also admissible? Would you prefer using h2 or using h3 in conjuction with A*? Give reasons for your anwers[4].

Yes, h3 is admissible. If h1 and h2 always underestimate the "true" cost then the lesser of the two will certainly underestimate the true cost as well; therefore, h3 is admissible.

I will prefer h2, because h2 is always greater equal than h3 and therefore it provides a closer approximation of the true cost. As a matter of fact, h2 dominates h3, which translates into equal or better efficiency of the search, as discussed on the bottom of page 106 of our textbook.

2) Local Search

a) Assume you apply randomized hill climbing to a minimization involving a continuous, differentiable function that has 3 minima. Will it always find the optimal solution? Give reasons for your answer! [3]

No, HC might climb down the wrong minimum depending on the chosen starting point.

There is also the possibility that it reaches a local minima, plateau or ridge that is not the same as the global minima.

b) What is the "main" difference between simulated annealing and randomized hill climbing? [2]
   SA does allow downward steps, meaning that we intentionally make bad moves and gradually decrease the consistency of those "bad" moves. A way that we can choose those bad moves is by randomly picking the next move out of a list of successor neighbors instead of the best next neighbor.

c) When does A* terminate? Be precise!

A* terminates when either the open list is empty or when a goal node is expanded. For the case where the goal node is reachable it means that when the goal node is removed from the open list and its neighbors are processed, A* concludes because it has found the shortest path to the goal.

5. Comparison of Seach Algorithms

Compare Traditional Hill Climbing/Randomized Hill Climbing and Best-first Search! What are the main differences between the two approaches?

	Randomized HC	Best First Search
The way they search	Explore a single path	Can explore multiple paths in parallel
	Only moves forward, cannot move backward	Jumps between states; can explore multiple paths in parallel!
Storage	O(1) / O(m)[1]	O(n)
Runtime	O(m) but might stop prematurely, fast	O(n) in the worst case, not fast
Finding solutions in finite search spaces	might terminate prematurely; might go into the wrong direction and get stuck	Yes
Find global optimum	no	no, terminates permaturely
Parameter Selection	Choosing neighbor hood size and sampling rate for RHC is challenging	Straight Forward
Incorperating Heuristics	Needs good state evaluation function	Need good state evaluation function
Other	RHC is a probabilistic algorithm (usually) returns different solutions in different runs	deterministic

Let n be the size of the search space, the number of nodes in the search tree
b the branching factor, the number of successors
m is the length of the longest path in the search tree

[1] Only if it is necessary to return the solution path, as in the 8-puzzle problem!

6) Constraint Satisfaction Problems [8]

Provide a definition[2] the letter constraint satisfaction problem given below:

1. Define the Variables
2. Define the set of values each variable can take
3. Define all constraints!

```
          T W O
    +     T W O
  ------------------------------
        F  O  U  R
```

Assume each letter can take only one digit, and reciprocally each digit can be associated to at most one letter.

Variables: T, W, O, F, U, R in {0…9}
X1, X2, X3 in {0,1}

Constraints:

1. DIFF(T, W, O, F, U, R)
2. O+O=R+10*X1
3. X1+W+W=U+10*X2
4. X2+T+T=O+X3
5. X3=F

[2] Be aware of the fact that you are not asked to solve this letter constraint satisfaction problem!

7) Discrete CSPs

Assume a constraint satisfaction problems in which variables A, B, C, D which take values in {1,…,100}

- **Constraints:**
    - (C1) A*B + B*C=D*D*D
    - (C2) A*D*D*D + A*C=B*B
    - (C3) B=A*D

A brute force solution to this problem could look as follows:

```
FOR A=1,…,A=100
FOR B=1,…,B=100
FOR C=1,…,C=100
FOR D=1,…,D=100 DO {
IF C1 and C2 and C3 THEN WriteSolution(A,B,C,D)}
```

Give the code of a more efficient solution to this problem which uses less loops and/or less iterations inside the loop. Briefly describe the idea of your solution!

One approach: Eliminate B

(C1') A*A*D +A*C*D=D*D*D

(C2') A*D*D*D+A*D=A*A*D*D

Faster Loop:

```
FOR A=1,…,A=100
FOR C=1,…,C=100
FOR D=1,…,D=100 DO {
IF C1' and C2' THEN {B=A*D; WriteSolution(A,B,C,D)}
```

Maybe enhance further … ---not needed for exam!

(C2") A*D*D + A = A*A*D

(C2"') D*D+1=A*D

(C1") A*A+A*C=D*D

combine the last two getting (4) A*D – 1=A*A + A*C

now the code, displayed above, can be further simplified by solely a single equation: (4)!