Midterm 2

Name: Targaryen

SID: 0123456789

Name and SID of student to your left: Lannister

Name and SID of student to your right: Stark

Circle One: Pimentel Wheeler Wozniak

Rules and Guidelines

- **The exam is out of 80 points and will last 80 minutes.**
- Answer all questions. Read them carefully first. Not all parts of a problem are weighted equally.
- Write your student ID number in the indicated area on each page.
- Be precise and concise. **Write in the solution box provided.** You may use the blank page on the back for scratch work, but it will not be graded. Box numerical final answers.
- Any algorithm covered in lecture can be used as a blackbox, unless otherwise stated.
- If no specific runtime is provided in the question, find as efficient an algorithm as you can. If a runtime is specified and you cannot achieve it, but you can get a weaker runtime, then submit that solution.
- Good luck!

Discussion Section [1 point]

Which of these do you consider to be your primary discussion section(s)? Feel free to choose multiple, or to select the last option if you do not attend a section. **Please color the checkbox completely. Do not just tick or cross the boxes.**

- ☐ Aarash, Tuesday 5 - 6 pm, Barrows 151
- ☐ Nick T., Tuesday 5 - 6 pm, Wheeler 202
- ☐ Chinmay, Wednesday 9 - 10 am, Dwinelle 215
- ☐ Chinmay, Wednesday 10 - 11 am, Moffitt 150D
- ☐ Simin, Wednesday 10 - 11 am, Mulford 240
- ☐ Aditya M., Wednesday 11 am - 12 pm, Barrows 56
- ☐ Nick W., Wednesday 11 am - 12 pm, Giannini 141
- ☐ Yuxiang, Wednesday 1 - 2 pm, Dwinelle 215
- ☐ Nikhil, Wednesday 1 - 2 pm, Wheeler 108
- ☐ James, Wednesday 1 - 2 pm, Soda 405
- ☐ Aditya B., Wednesday 2 - 3 pm, Wheeler 200
- ☐ Owen, Wednesday 2 - 3 pm, Etcheverry 3105
- ☐ James, Wednesday 2 - 3 pm, Soda 310
- ☐ Aditya B., Wednesday 3 - 4 pm, Wheeler 202
- ☐ Harley, Wednesday 3 - 4 pm, Wheeler 220
- ☐ Michael, Wednesday 3 - 4 pm, Soda 310
- ☐ Vinay, Wednesday 4 - 5 pm, Dwinelle 223
- ☐ Benjamin, Wednesday 5 - 6 pm, GPB 107
- ☐ Mudit, Wednesday 5 - 6 pm, Moffitt 150D
- ☐ I do not attend a discussion section

1 Multiple Choice [2 points per problem]

Fill in a single square for each problem. Fill it in completely.

(a) Say we run the Huffman encoding algorithm on a file with n *distinct* symbols, each of which appears an equal number of times. What are the lengths of the longest and shortest resulting codewords? (Assume that n is a power of 2)

- ☐ Longest $n-1$; shortest 1
- ☐ Longest $\log(n)$; shortest $\log(n)$
- ☐ Longest $\frac{n}{2}$; shortest $\frac{n}{2}$
- ☐ Longest n; shortest $n-1$

Solution: Longest $\lceil \log(n) \rceil$; shortest $\lfloor \log(n) \rfloor$. The difference in the the depths of two symbols in the Huffman encoding tree cannot be more than 1, so the encoding tree will end up looking like a binary tree.

(b) Say we had a similar setup to the last part, except that symbol i appears 2^i times ($0 \leq i \leq n-1$). Now what are the lengths of the longest and shortest codewords? (Assume that n is a power of 2)

- ☐ Longest $n-1$; shortest 1
- ☐ Longest $\log(n)$; shortest $\log(n)$
- ☐ Longest $\frac{n}{2}$; shortest $\frac{n}{2}$
- ☐ Longest n; shortest $n-1$

Solution: Longest $n-1$; shortest 1. Symbol $n-1$ will appear at the top level of the encoding tree, symbol $n-2$ will appear at the second level, and so on. Each level of the encoding tree will have 1 symbol on it, except that the last level will have 2.

(c) Suppose we have a connected graph G on n vertices such that three of the edges in G have weight 0 while all other edges have distinct positive weights (ie, other than the original three, no pair of edges can have the same weight). What is the maximum number of MSTs G could have? What is the minimum number?

- ☐ Maximum 1; minimum 1
- ☐ Maximum $\lfloor \frac{n}{3} \rfloor$; minimum 1
- ☐ Maximum 3; minimum 1
- ☐ Maximum 3; minimum 3

Solution: Maximum 3; minimum 1. If the three weight zero edges do not create a cycle, we have to include all of them in the MST–and since all other edges have distinct weights, once we've decided which of the weight zero edges to include, there is only one choice for how to finish constructing the MST. If the tree weight zero edges form a cycle, any MST must include exactly two of them. Once we've chosen which of the two to include, we again have that there is only one choice for how to construct the rest of the MST since all other edge weights are distinct. Thus, we have 1 MST in the former case and 3 in the latter case; since these two cases cover all possibilities, we have that these are respectively the minimum and maximum values.

(d) Consider the following Horn SAT formula on five variables:

$$(c \wedge e) \Rightarrow a,\ (a \wedge c) \Rightarrow d,\ \Rightarrow a,\ e \Rightarrow c,\ a \Rightarrow b,$$
$$(\overline{a} \vee \overline{d} \vee \overline{e})$$

How many satisfying assignments are there for this formula?

- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3

Solution: 3. Running the Horn SAT greedy algorithm, we see that a and b both must be true in any satisfying assignment. If we also set c to true, the implication $(a \wedge c) \implies d$ means that d must also be true. From the negation $(\overline{a} \vee \overline{d} \vee \overline{e})$, we then get that e must be false. Thus, setting c to true gives only one satisfying assignment. If instead we set c to false, the implication $e \implies c$ tells us that e must also be set to false. We can then set d either to true or to false, giving us two additional solutions.

(e) Suppose we have some instance of the Knapsack problem. Let U be the value of the optimal solution if there is an unlimited number of each item available and V be the optimal value if there is only one of each item available. What is the strongest relationship we can give between U and V?

- ☐ $U \leq V$
- ☐ $U \geq V$
- ☐ $U = V$
- ☐ None of the above are guaranteed to be true.

Solution: $U \geq V$. Let S_V be the optimal solution to the problem where there is only one of each item available. Since the items in this problem are the same as the items in the problem where there is an unlimited supply of each item, S_V is also a valid collection of objects to take in this latter problem. If S_V is also an optimal solution to the unlimited supply question, we have that $V = U$. If not, it must be because there is some solution S_U to the unlimited supply problem such that the objects in S_U are worth more than the objects in S_V, meaning $U > V$. Since these are the only two possibilities, we have that $U \geq V$.

For the next 2 parts, let $G = (V, E, w)$ be an undirected weighted graph with *distinct* edge weights w. Let $G' = (V, E, w')$ be the same graph with weights defined by $w'(e) = w(e) + 1$.

(f) Suppose you ran Prim's Algorithm on G and G'. Would the spanning trees produced be the same?

- ☐ Always
- ☐ Never
- ☐ Depends on the graph G

Solution: Always. The same implementation of Prim's will start with start with the same vertex, and then pick the smallest edge going out of it. This will be the same edge as before. Now assume that Prim's has formed a partial MST which is correct. In the next step, Prim's will pick the smallest edge across the cut formed by the vertices in the partial MST and all the other vertices. Since all edge weights were unique, increasing every edge's weight by one will still maintian uniqueness and still maintain the ordering of edges. The same edge as before will be picked. This completes the argument.

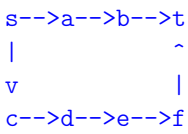
(g) Now suppose you ran Dijkstra's Algorithm on G and G' for vertices $s, t \in V$. Would the shortest path produced be the same?

- ☐ Always
- ☐ Never
- ☐ Depends on the graph G

Solution: Depends on the graph G. To come up with an example of a graph where the shortest path will be the same, simply consider a graph such as:

```
s---a---b---c---d----t.
```

To consider a graph where the shortest path is not the same, consider the following:



where s->a is 0.1, a->b is 0.2, b-t is 0.3, s->c is 0.05, c->d is 0.04, d->e is 0.03, e->f is 0.02, f->t is 0.01.

In this case, the shortest path from s to t is $s - c - d - e - f - t$ of length 0.15.

After adding 1 to each edge, the shortest path from s to t is $s-a-b-t$ of length 3.6, since $s-c-d-e-f-t$ would have length 5.15.

2 "The Final Problem" [10 points]

Sherlock Holmes boards the train from London to Dover in an effort to reach the continent and so escape from Professor Moriarty. Moriarty can take an express train and catch Holmes at Dover. However, there is an intermediate station at Canterbury at which Holmes may detrain to avoid such a disaster. But of course, Moriarty is aware of this too and may himself stop instead at Canterbury. Von Neumann and Morgenstern once estimated the value to Moriarty of these four possibilities to be given in the following matrix (in some unspecified units).

		Holmes	
		Canterbury	Dover
Moriarty	Canterbury	200	-100
	Dover	0	200

Assume Moriarty goes to Canterbury with probability x_1 and Dover with probability x_2 for x_1, x_2 known to Holmes. What is the expected payoff to Moriarty assuming Sherlock plays optimally? Formulate this as a minimization linear program. You do not need to express the linear program in canonical form.

$$\text{OPT}_{\text{row}} = \begin{cases} \max \\ \\ \text{s.t.} \\ \\ \\ \\ \end{cases}$$

Solution: For given x_1 and x_2, the column player chooses $\min\{200x_1, -100x_1 + 200x_2\}$, which the row player wants to maximize over feasible x_1 and x_2. (Note how the first three lines below are equivalent to $\max\{200x_1, -100x_1 + 200x_2\}$.)

$$\begin{aligned} \max\ & w \\ & w \leq 200x_1 \\ & w \leq -100x_1 + 200x_2 \\ & x_1 + x_2 = 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

What are the optimal strategies for Holmes and Moriarty, and what is the value of the game? (Historically, as related by Dr. Watson in *The Final Problem* in Arthur Conan Doyle's *The Memoires of Sherlock Holmes*, Holmes detrained at Canterbury and Moriarty went on to Dover.)

Answer:

Solution: The row player wants to choose x_1 and x_2 which maximizes $\min 200x_1, -100x_1 + 200x_2$. The maximum will occur when both quantities are equal to each other. So, we have

$$200x_1 = -100x_1 + 200x_2 \implies 300x_1 = 200x_2 \implies 3x_1 = 2x_2$$

. Combining that with $x_1 + x_2 = 1$, we get $x_1 = 0.4$, $x_2 = 0.6$.

Similarly, the column player wants to choose y_1 and y_2 which minimizes $\max 200y_1 - 100y_2, 200y_2$. The maximum will occur when both quantities are equal to each other. So, we have

$$200y_1 - 100y_2 = 200y_2 \implies 200y_1 = 300y_2 \implies 2y_1 = 3y_2$$

. Combining that with $y_1 + y_2 = 1$, we get $y_1 = 0.6$, $y_2 = 0.4$.

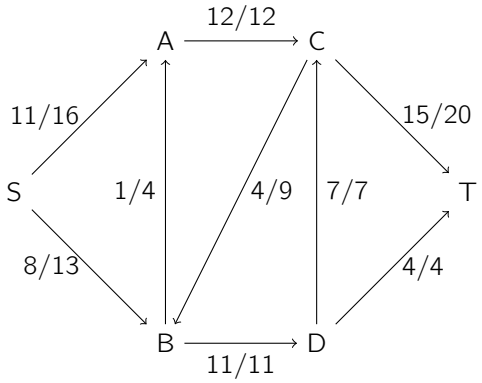
The value of the game is defined as the expected payoff for the row player, which is

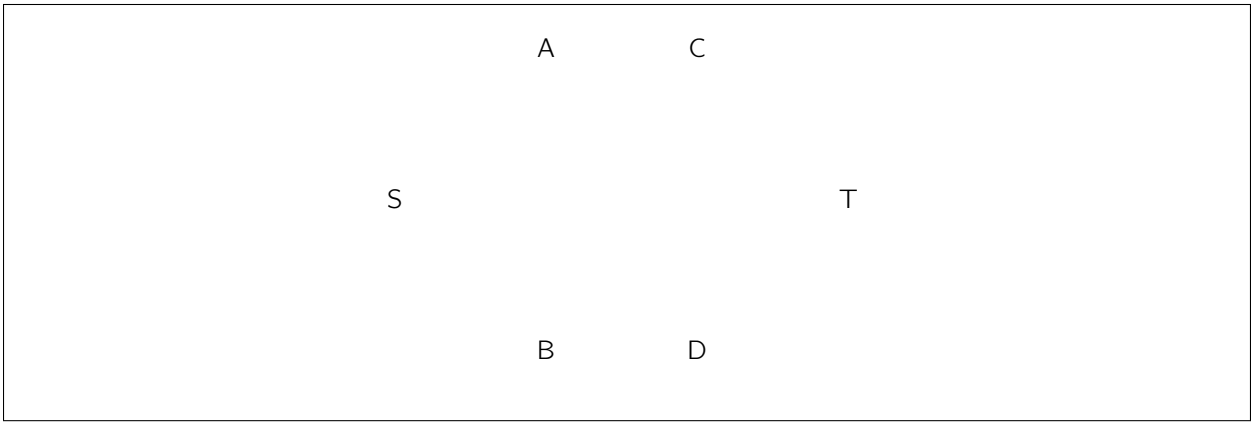
$$200x_1y_1 - 100x_1y_2 + 0x_2y_1 + 200x_2y_2 = 200(0.24) - 100(0.16) + 200(0.24) = 80$$

3 Computing Flows [15 points]

Suppose you run Ford-Fulkerson on the given graph to compute the max-flow from vertex A to vertex F. After several iterations you end up with following flow (an edge labelled "x/y" denotes an edge of capacity y with x flow going through it):

Draw the residual graph for this flow.





What is an augmenting path that the Ford-Fulkerson algorithm could pick on its next iteration. How much flow would be sent along this path?

Draw the new residual graph after executing the iteration described in the previous part.

A C

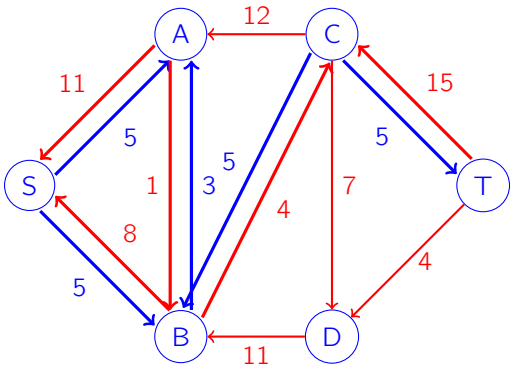
S T

B D

Will Ford-Fulkerson execute for another iteration? If so, detail another path it could pick and the flow along it. If not, state the min-cut of the graph.

Solution:

a The solution is as follows (back edges in red):



Please note that since this question is testing whether or not students understand what a residual graph is and how to draw it, students who didn't draw the graph correctly received no points. For instance, drawing only the back edges was given no points. Partial credit was only awarded rarely, and in cases where the grader believed that the error was purely due to copying down a lot of numbers.

b **Solution A:** Pick the path S-B-C-T. Send 4 units of flow along it.

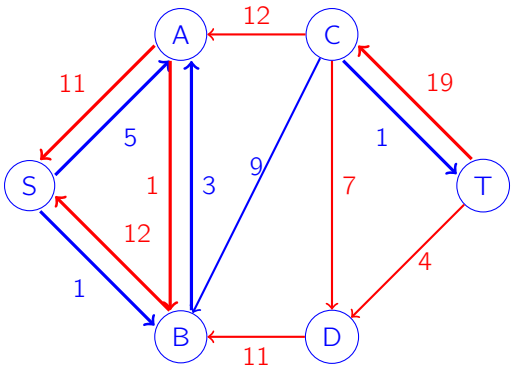
Solution B: Pick the path S-A-B-C-T. Send 1 unit of flow along it.

Credit was awarded if part (a) was incorrect, but the path picked was correct based on part (a). Note that a **path** must be specified to push flow, not just an *edge*.

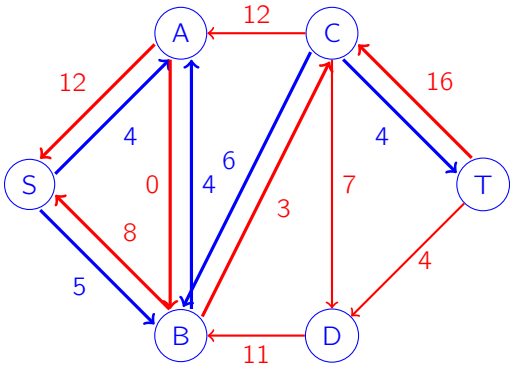
A common mistake in part (a) was students drawing the edges between B and C reversed (i.e., 5 going from B-C and 4 from C-B.) If this was the case, many students wrote that the path picked is S-B-C-T and 5 units of flow was sent. This was given full points for this part, since it is correct based on an incorrect previous part.

c The solution is as follows:

Solution A:



Solution B:



For this part too – points were awarded based on the solution that the student wrote in part (b). However, since this part tests whether students know how to draw residual graphs or not too (like part (a)), no points were awarded if a student did not draw all the edges, including forward edges. For most submissions that got part (a) wrong, they got part (c) wrong too.

d **Solution A:**
One min cut of the graph is to partition S,A,B into one set, and put C,D,T in the other.

Note that a cut is fully specified by two **sets of vertices**. A "min-cut" is not the *value* of the sum of the edge capacities going across it, but is the *cut* itself. Students who simply stated the value of the sum of the edge capacities going across the cut received minimal partial credit for observing that the algorithm cannot continue. A cut is also not described by *edges*. For instance, the set $\{AC, BD\}$ does **not** describe the min-cut.

Also, for the min-cut, S and T **must** be in different sets. Students who put S and T in the same set received 0 points. The two sets must also be disjoint and placing vertices in both sets resulted in 0 points.

Solution B:
Another path that the algorithm could pick is S-B-C-T, and it could send 3 units of flow down this path.

4 Duality [10 points]

Fill in the boxes for the dual $(\mathcal{D})$ to the following linear program $(\mathcal{P})$.

$$(\mathcal{P}) = \begin{cases} \max & 2x_1 + x_2 \\ \text{s.t.} & x_1 + 2x_2 \leq 5 \\ & 3x_1 + 4x_2 \leq 2 \\ & x_1, x_2 \geq 0 \end{cases} \qquad (\mathcal{D}) = \begin{cases} \min & \square\, y_1 + \square\, y_2 \\ \text{s.t.} & \square\, y_1 + \square\, y_2 \geq \square \\ & \square\, y_1 + \square\, y_2 \geq \square \\ & y_1, y_2 \geq 0 \end{cases}$$

Show that *any* feasible solution of the dual provides an upper bound to *any* feasible solution of the primal. You may either write a couple of sentences arguing how the construction of the dual program necessarily implies this, or actually work through the math to prove it.

Solution: There were broadly 3 acceptable answer "types" for part b. The first is an argument for why the construction of the dual guarantees its feasible points will upper bound the primal's feasible points. The second is an argument from strong duality, and the third is a very direct proof using linear algebra. An example of what would be acceptable for each is detailed below.

1 We constrain y_1, y_2 so that they yield coefficients to x_1, x_2 that are >= than those in the primal objective. Specifically: our linear combination will give us the end result $(y_1 + 3y_2)x_1 + (2y_1 + 4y_2)x_2 \leq 5y_1 + 2y_2$. The constraints in our dual force $y_1 + 3y_2 \geq 2$ and $2y_1 + 4y_2 \geq 1$ – and furthermore since x_i and y_i are strictly non-negative, this ensures that $2x_1 + x_2 \leq 5y_1 + 2y_2$ for all feasible combinations of x_i's and y_i's.

2 Alternatively, we know that the optimal solutions to the primal and dual will achieve the same value (either invoke strong duality on LP's or find them directly using the graphical method to show this). We know all feasible solutions to the dual are >= this optimum and all feasible solutions to the primal are <= this same optimum. Thus, any feasible solution to the primal is <= any feasible solution to the dual.

3 The final approach is detailed here (the series of inequalities under theorem 4.1 was pretty much all that was needed). Very few students used this solution.

Common Mistakes:
In attempting the first solution, many students simply wrote "$(y_1 + 3y_2)x_1 + (2y_1 + 4y_2)x_2 \leq 5y_1 + 2y_2$," without justifying why this in turn leads to an upper bound on $2x_1 + x_2$.

In attempting the second solution, many students made a loose argument about minimization and maximization of the dual and primal respectively without mentioning that their optimums achieve the same value.

Conversely some students proved that the optimums achieved the same value without relating this fact to the question asked.

A very small amount of points (.5) were deducted for students who attempted the first solution but did not mention the significance of the non-negativity of the x_i's and y_i's.

Any combination of these partial solutions was worth no more than one of them individually.

No points were deducted for spelling "dual" as "duel".

5 Fixed leaf MST [15 points]

Given an *undirected* weighted graph $G = (V, E)$ and a subset $S \subseteq V$ of the vertices, devise an algorithm that generates a spanning tree of minimum weight such that each vertex $s \in S$ is a leaf of the tree. Note that the tree is allowed to have additional leaves not in S. Recall that a leaf of a tree is a vertex with degree 1. If it is not possible to construct such a spanning tree, then your algorithm should return "NOT POSSIBLE" or some similar indicator. For partial credit, solve the case when $|S| = 1$.

Provide just your main idea and a brief justification. For full credit your algorithm should be as efficient as possible.

Solution:

Solution:
Solution 1: First, find the MST of $V \setminus S$ using Prim's or Kruskal's. Then, for every $s \in S$, add the lowest-weight edge connecting s to a vertex in $V \setminus S$. If no MST of $V \setminus S$ exists, or if any vertex $s \in S$ has no edge to $V \setminus S$, it is impossible.
Solution 2: Run a modified version of Kruskal's algorithm, skipping any edge that either (1) connects two vertices of S, or (2) is incident to a vertex of S that is already in the tree.

Common mistakes:

- Adding the lightest edge incident to each vertex $s \in S$, but not ensuring that it doesn't connect to another vertex in S

6 Successful passage through Gridworld [15 points]

You have been flattened into a 2D occupant of Gridworld, which is a $n \times n$ sized grid. Your goal is to move from $(0, 0)$ to (n, n). You are given a map of Gridworld, the matrix $G \in \mathbb{Z}^{n \times n}$, and know that while traveling *out* of any given cell, you will either pay tokens for passage or be gifted tokens by a generous stranger. The value you will lose/gain at (i, j) is recorded in $G(i, j)$; $G(i, j)$ is positive if you gain tokens. You can only continue moving on the grid when you have ≥ 0 tokens, and you may move from (i, j) to $(i + 1, j)$ or $(i, j + 1)$.
Give a dynamic programming algorithm to compute the minimum number of tokens you need to begin with (at (0,0)) to ensure successful passage.

Define your subproblem(s):

Solution: Let $T(i, j)$ be the minimum number of tokens needed to start at cell (i, j) and reach the terminal cell (n, n).

What are the base cases?

Solution: $T(n, n) = 0$. Another acceptable answer is $T(n, n) = \max\{0, -G(n, n)\}$. Note $T(i, n)$ and $T(n, j)$ are not base cases since they have subproblems of $T(i + 1, n)$ and $T(n, j + 1)$ but no points were penalized for this.

Write the recurrence relation for the subproblems.

Solution: Let $T(i,j) = \max\{\min\{T(i+1,j), T(i,j+1)\} - G(i,j), 0\}$.
We think of the tokens needed to get from (i,j) to (n,n) as the sum the tokens needed to exit the cell ($-G(i,j)$, since you need positive number of tokens to begin with if you are going to pay (i.e. if $G(i,j)$ is negative)) and the tokens needed to get from your next cell to (n,n). We take the minimum over the various next cells that we can choose. Further, we need to take the maximum between this value and 0 because we can never assume we start with negative coins at a cell. For an example of where this max matters, we might start at cell s and go into debt before reaching a cell t that has negative T() value (meaning you can start with 0 coins and finish with a positive number of coins at that square), so s should have a positive T() value, but if the max was not there, the negative value from t would make the T() value of s negative.

What is the time complexity of your algorithm in Big Theta notation? What is the best possible space complexity you can achieve? Briefly explain.

Solution: Runtime is $\Theta(n^2)$ because there are n^2 problems that each look at only 2 subproblems, and therefore each take constant time.
Space complexity is $\Theta(n)$ because we only need to remember the previous row or column as we fill out the current row or column.

Blank page: If using this page to write an answer, clearly specify on the question that your work should be found here.

Blank page: If using this page to write an answer, clearly specify on the question that your work should be found here.

.