M19CSE483 End-sem exam

Shubodh Sai P

TOTAL POINTS

18 / 40

QUESTION 1

1 Q1 - Rover navigation stack 3 / 5

+ **1 pts** Localization
+ **2 pts** Mapping
+ **2 pts** Planning
+ **2.5 pts** Partial in each
✓ **+ 0.5 pts** **partial localization**
✓ **+ 1 pts** **partial Mapping**
✓ **+ 1 pts** **partial planning**
+ **5 pts** correct
+ **0 pts** Not correct
+ **1 pts** partial info in [any 1/any 2/all] of the 3 asked.
✓ **+ 0.5 pts** **partial info in [any 1/any 2/all] of the 3 asked**

QUESTION 2

2 Q2 - Bundle adjustment 2 / 5

✓ **+ 1 pts** **part (a)**
+ **0.5 pts** part (a) partial
+ **1 pts** part (b)
+ **0.5 pts** part (b) partial
+ **1 pts** part (c)
+ **0.5 pts** part (c) partial / other partial
✓ **+ 1 pts** **part (d) 1**
+ **0.5 pts** part (d) partial 1
+ **1 pts** part (d) 2 / part 2 whole partial
+ **0.5 pts** part (d) partial 2
+ **0 pts** incorrect

QUESTION 3

3 Q3 - Position estimation 0 / 5

+ **5 pts** Complete
+ **2 pts** Right approach
+ **1 pts** Cost function
+ **1 pts** Jacobian
+ **1 pts** Algorithm
✓ **+ 0 pts** **Incorrect / Not attempted**

QUESTION 4

4 Q4 - 3D Extended Kalman Filter 3 / 10

✓ **+ 1 pts** **State vector and dimensions**
+ **2 pts** Motion model
✓ **+ 2 pts** **Observation model**
✓ **+ 1 pts** **Observation noise covariance dimensions**
+ **2 pts** Kalman gain dimensions
+ **2 pts** Observation model Jacobian
+ **0 pts** Incorrect / Not attempted
- 1 Point adjustment

R,t in observation model should be explained more clearly

QUESTION 5

5 Q5 - Essential matrix with motion constraints 3.5 / 5

✓ **+ 1 pts** **Computed translation**
+ **0.5 pts** Computed translation partially
✓ **+ 1 pts** **Computed rotation**
+ **0.5 pts** Computed rotation partially
+ **1 pts** Computed E-matrix
✓ **+ 0.5 pts** **Computed E-matrix partially**
✓ **+ 1 pts** **Correct number of parameters with explanation**
+ **0.5 pts** Correct number of parameters
+ **1 pts** Correct number of points with explanation
+ **0 pts** Incorrect / Not attempted

QUESTION 6

6 Q6 - RRT 3 / 5

✓ **+ 3 pts** **BasicRRT algo**

+ **1 pts** basicRRT- problem and solution
+ **1 pts** GoalbiasedRRT - problem and solution
+ **0 pts** Not attempted/Incorrect

BasicRRT can find a path to goal if enough iterations are given. Whereas GoalbiasedRRT will get stuck.

QUESTION 7

7 Q7 - Basic MPC **3.5 / 5**

✓ **+ 1 pts** **Cost function**

✓ **+ 1 pts** **Constraints**

✓ **+ 1 pts** **Obstacle avoidance constraint**

+ **2 pts** Linearizec obstacle avoidance constraint
+ **0 pts** Not attempted/Incorrect

+ 0.5 **Point adjustment**

0.5: d) Missing steps

CSE483-Mobile Robotics

End-semester exam

Monsoon 2019

November 18th

Maximum points: 40

Duration: 180 minutes

Instructions

- This is an open-book exam. You are allowed to use any paper notes or textbooks that you have brought with you.
- Laptops, tablets, or smartphones are NOT allowed. You also cannot collaborate with other students.
- Your answers must be concise and to-the-point. Verbosity will NOT fetch you additional marks.
- Sufficient space has been provided for each question. Using additional sheets are discouraged, if you need them you're probably doing something wrong.
- You do NOT get credit for replicating whatever is present in the textbook or your notes. Please do not fill your answer scripts with excerpts from such sources.
- Use the last page for rough work or for any of your answers, if necessary.
- State your assumptions clearly if there is any ambiguity with the question(s).

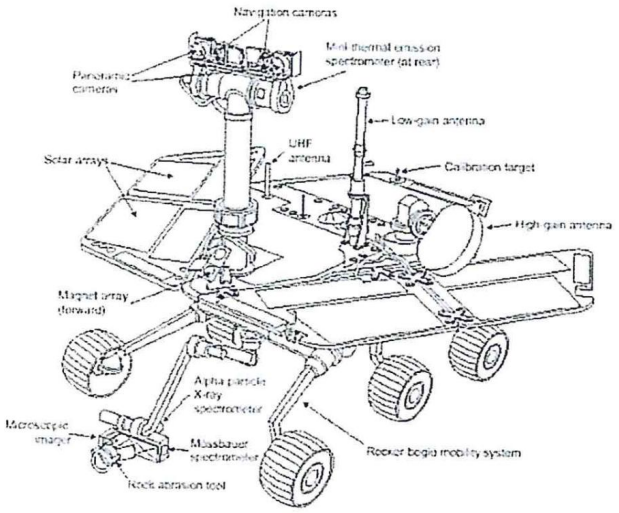
Roll number: 2019701013

Seat: C25

Invigilator sign:

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Total

Q1) A mars rover, equipped with cameras and various scientific instruments, is tasked to autonomously explore the surface of mars and collect data. Ignoring all the scientific instruments, and considering only the stereo navigation camera, systematically describe in words how you'd implement the localization, mapping, and planning modules to achieve this task. A high-level description of the algorithms in sufficient. (5 points)



Mars Exploration Rover

Using a stereo navigation system, we will be able to obtain the depth & disparity corresponding to every 3D point in the world.

I can now apply 8 point algorithm & further decomposition to get (R, t) b/w current & previous frame as the robot moves. using say, left images of stereo camera. (or apply stereo rectification & take both!)

LOCALIZATION:

Now, localization is done but it might not be upto scale. It can be scaled using the depth from stereo camera. I use SIFT/SURF algorithm for doing feature matching & correspondences, now that I have correspondences, I can use triangulation to find the 3D point corresponding to a pixel in 2nd image.

MAPPING: Which means I can map the points. After registering the 3D points of two point clouds, I can use Iterative Closest Point (ICP) for further refinement. Now, SLAM is being done correctly, I have a global map

PLANNING: out of which I can make an occupancy grid, 0 → free space, 1 → obstacle

& apply a planning algorithm like Dijkstra / A^* to navigate. If it is a non-holonomic robot, we can do RRT / Model Predictive Control for better planning.

Q2) Consider two calibrated cameras C_1, C_2 that are looking at a scene. The relative orientation R, t between the two cameras, and an up-to-scale 3D point cloud of the scene, $\{X_i\}$, were estimated by applying our well known linear 8-point and triangulation algorithms. We now wish to refine these estimates by applying bundle adjustment. (a) Write down the cost function to minimize. (1 point) (b) Why is the cost function squared? (1 point) (c) How can the cost function be written in linear form? (1 point) (d) What are the dimensions of the Jacobian? Show the structure of the Jacobian in block form. (2 points)

(a) $\lambda_1 \vec{P}_1 = K_1 \vec{X}$ $\quad \lambda_2 \vec{P}_2 = \vec{K}_2 (R\,t)\, \vec{X}$

Given calibrated, so K_1 & K_2 known. we want to optimize over (R, t) & X_i.

Although K is known writing $\vec{K}_2 (R\,t)$ as a matrix P_2, the cost function would be.

$$\vec{X}_j = \begin{bmatrix} x_j \\ y_j \\ z_j \end{bmatrix} \qquad P_2 \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{14} \\ P_{31} & P_{32} & & P_{34} \end{pmatrix}$$

$$l = \left(\frac{P_{11} X_j + P_{12} Y_j + P_{13} Z_j + P_{14}}{P_{31} X_j + P_{32} Y_j + P_{33} Z_j + P_{34}} - \hat{x}_j \right)^2 + \left(\frac{P_{21} X_j + P_{22} Y_j + P_{23} Z_j + P_{24}}{P_{31} X_j + P_{32} Y_j + P_{33} Z_j + P_{34}} - \hat{y}_j \right)^2$$

$(\hat{x}_j, \hat{y}_j)$ → Predicted projection of j point to 2nd camera

(x_j, y_j) → observation of j^{th} pixel in 2nd camera

(b) the cost function is "l_2" norm. $\|x\|^2$ We take this norm as

(1) It is convex, so easy to optimize.

(2) Squaring will give the loss value same even if ordering is different. $(a-b)^2 = (b-a)^2$

(c) We can take "l_1" norm (absolute value) .. i.e $|\hat{x}_i - x_i| + |\hat{y}_i - y_i| = l$.

(d) NOTE:

I am optimizing only over 2nd camera as first P only consists of K which is known. So, $J_A \rightarrow (2 \times 15)$ in case when not optimizing over P_1 (first camera)

However, if we still want to optimize over K_1, J_B would be (4×27).

$(2nm \times (12m + 3n))$

Showing for J_A (15 columns)

$$2\begin{cases} = \begin{bmatrix} \vec{J}^{m}_{11\,(1\times m)} & \vec{J}^{s}_{11\,(1\times 3)} \\ \vec{G}^{m}_{11\,(1\times 12)} & \vec{G}^{s}_{11\,(1\times 3)} \end{bmatrix} \end{cases}$$

J_B would be (27 columns)

$$2\begin{cases} = \begin{bmatrix} \vec{J}^{m}_{11\,(1\times 12)} & 0 & \vec{J}^{s}_{11\,(1\times 3)} \\ G^{m}_{11} & 0 & G^{s}_{11\,(1\times 3)} \\ 0 & \vec{J}^{m}_{21\,(1\times 12)} & \vec{J}^{s}_{21} \\ 0 & \vec{G}^{m}_{21} & \vec{G}^{s}_{21} \end{bmatrix} \end{cases}$$

$$\vec{J}^{m}_{ij} = \left(\frac{\partial f_n(\vec{P}_i, \vec{X}_j)}{\partial \vec{P}_i} \right)_{1\times 12} \quad ; \quad \vec{J}^{s}_{ij} = \left(\frac{\partial f_n(\vec{P}_i, \vec{X}_j)}{\partial \vec{X}_j} \right)_{1\times 3}$$

$$\vec{G}^{m}_{ij} : \frac{\partial f_y \ldots}{\ldots} \quad ; \quad \vec{G}^{s}_{ij} : \left(\frac{\partial f_y \ldots}{s} \right)_{1\times 3}$$

Q3) Suppose a 2D robot exists in space and it measures its distance from a set of n known landmarks located at $l_1, l_2, ..., l_n$ using a range sensor. Each distance measurement r_i is given as $r_i = \|x - l_i\| + \eta_i$, where x is the robot location and η_i is some unknown noise. Given these noisy measurements, describe in detail how you'd estimate the location of the robot. Derive any intermediate quantities needed. (5 points)

Assumption: Let's say we have initial location $\bar{\mu}_t$ & $\bar{\Sigma}_t$ & $Q_t = \begin{pmatrix} \text{min} & 0 \\ 0 & \text{bar} \end{pmatrix}$

$$r_i = \sqrt{(x_k - l_{xi})^2 + (y_k - l_{yi})^2} \qquad \text{Let } (x_k - l_{xi})^2 + (y_k - l_{yi})^2 = M$$

$$\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$$

($\bar{\mu}_t$: assumed ✓; K_t: ?; z_t: sensed by sensor, so given ✓; $h(\bar{\mu}_t)$: r_i ✓)

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$$

($\bar{\Sigma}_t$ ✓, H_t^T ?, H_t ✓, Q_t ✓)

$$H_t = \begin{pmatrix} \frac{\partial r_i}{\partial x_k} & \frac{\partial r_i}{\partial y_k} \end{pmatrix} = \begin{bmatrix} \frac{2(x_k - l_{xi})}{2\sqrt{M}} & \frac{2(y_k - l_{yi})}{2\sqrt{M}} \end{bmatrix}$$

$$H_t^i = \begin{bmatrix} \frac{2(x_k - l_{xi})}{\sqrt{M}} & \frac{(y_k - l_{yi})}{\sqrt{M}} \end{bmatrix} \text{ where } M \text{ is}$$

So, we know, H_t, we can find out K_t; using K_t, we can find out μ_t ✓ ②

$$\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$$

K_t, H_t, $\bar{\Sigma}_t$ known, so, Σ_t can be calculated ✓ ①

① & ② give the state of robot.

ab.

Q4) A sensor platform, consisting of a stereo camera and an inertial measurement unit (IMU), moves freely through a 3D scene. We identify the pose of the platform as the pose of the IMU. The rotation and translation of the camera relative to the IMU is known and is given by R_c^I and t_c^I respectively. We wish to estimate the pose of the platform with respect to some fixed world frame, using the sensor measurements as the platform moves around via an extended Kalman filter.

The scene contains known point landmarks on the floor which are observed by the stereo camera, as $y_k = [u_{left}, v_{left}, u_{right}, v_{right}]^T$, for every landmark at every time step k. The 3D locations of these landmarks are known in the world frame. The angular velocity and translational velocity of the platform, ω_k, v_k, relative to the IMU, are obtained from the IMU measurements for every time step k.

(a) What is the state vector and its dimension? **(1 point)** (b) Derive the translational and rotational motion models for the platform. (*Hint:* $\Psi_k = \omega_k dt$, where Ψ_k is the angle-axis representation for the rotation.) **(2 points)** (c) Derive a suitable observation model for the stereo camera. **(2 points)** (d) What are the dimensions of the observation noise covariance? **(1 point)** (e) What are the dimensions of the Kalman gain for a single landmark? **(2 points)** (f) Derive the Jacobian for the observation model. **(2 points)**

$x(\vec{P}) \to K(Rt)\vec{X}$

IMU $\to R_c^I, t_c^I$

? → unknown
✓ → known

(a) State vector: $(x, y, z, \text{roll}, \text{pitch}, \text{yaw})$. Dimension 6.

(c) $z_p = \frac{bf}{u_{left} - u_{right}}$

$X = z_p$

Camera eqn: $\vec{P} = K(Rt)\vec{X}$ (given; world to camera; known in world frame)

z_p known for landmark

obtained from motion model. (for now, assume "b" is solved; explained here.)

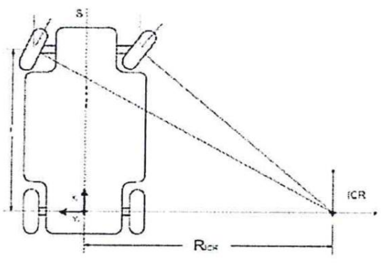
$P_{left} = \begin{bmatrix} u_{left} \\ v_{left} \end{bmatrix} = K(Rt)\vec{X}$ known

$\begin{pmatrix} u_{left} \\ v_{left} \\ u_{right} \\ v_{right} \end{pmatrix} = \begin{bmatrix} K(Rt)_{left}\vec{X} \\ K(Rt)_{right}\vec{X} \end{bmatrix}$ (assume "b" is solved)

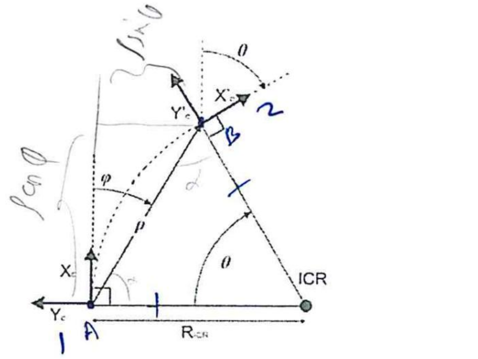
(d) Dim(obs noise covar) = 4 × 4.

(e) 2 × 4

Q5) Consider a non-holonomic car that is equipped with a camera. The camera axes are denoted by X_c and Y_c, and is placed such that X_c is perpendicular to the rear-wheel axis as shown in figure (a). Due to the motion constraints of the car, the camera's motion can be perceived to be a circular motion, as shown in figure (b). X_c' and Y_c' are the camera axes in motion, whose origin with respect to the first frame, in polar coordinates, is at ρ, ϕ and is rotated by θ.



(a)



(b)

(a) Compute the essential matrix E for this configuration. **(3 points)** (b) How many parameters does it have? How many corresponding image points are required for estimating this E? It is not eight. **(2 points)**

$R_2^1 = \begin{bmatrix} \cos A & -\sin A \\ \sin A & \cos A \end{bmatrix}$

$A = -(\theta+\varphi)$ from (1) $= \begin{bmatrix} \cos(\theta+\varphi) & \sin(\theta+\varphi) \\ -\sin(\theta+\varphi) & \cos(\theta+\varphi) \end{bmatrix}$

$t_2^1 = \begin{bmatrix} \rho\cos\varphi \\ -\rho\sin\varphi \end{bmatrix} = \begin{bmatrix} \rho\cos\theta/2 \\ -\rho\sin\theta/2 \end{bmatrix}$ from (I)

$E = (t_2^1)_\times R_2^1$, $(t_2^1)_\times$ is skewsymmetric form of t_2^1

~~It has 3 parameters but as $\theta = 2\varphi$~~ thus 2 parameters (θ or ρ).

7 image points as there is a constraint

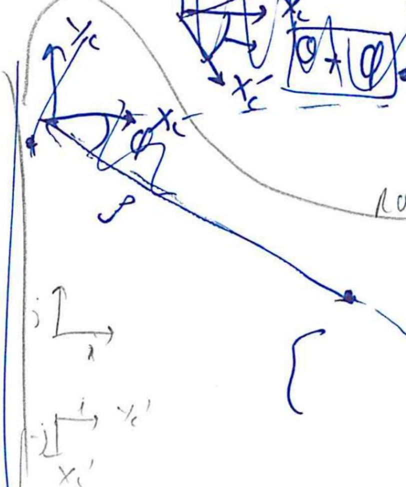
~~$2\alpha+\theta = 180$~~

$2\alpha + 2\varphi = 180°$ → (I)

$\theta = 2\varphi$

Angle b/w two frames

$\theta + \varphi$ → (1)



ROUGH

Q6) (a) Consider a holonomic disc-shaped robot with diameter, say d. The robot is initially at start location and is required to reach the goal location while avoiding obstacles. In order to avoid collision, the robot must maintain at least l distance (where $l = d/2$) from its center to the obstacle surface. It considers to have reached the goal if its center coincides with the goal location. Write an RRT (unidirectional, and not goal biased) algorithm for the robot to find a path from start to goal. **(3 points)** (b) State the likely problems faced by basic RRT and goal biased RRT in the given environment. Also suggest improvements to be done to the algorithm to resolve these problems. **(2 points)**

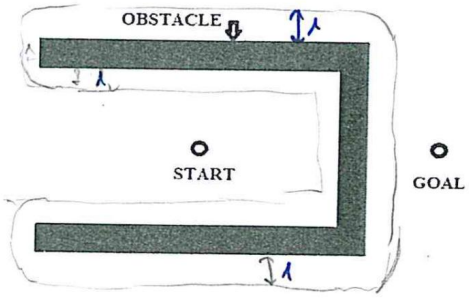


Figure 2: Robot environment

ALGORITHM

(a) RRT_ALGO(x_{init}) : (i) T.init(x_{init})

(ii) Inflate the obstacle by l (as shown in fig 2) as we treat it as point object. : obstacle space = X_{obs}

(iii) for k = 1 to k do

if collision free: $x_{rand} \in$ RANDOM_STATE();

EXTEND(T, x_{rand}) if collision free: check if it is belonging to space X_{obs}.

Return T

EXTEND(T, x):

(i) $x_{near} \leftarrow$ NEAREST_NEIGHBOUR(x, T);

(ii) if NEW_STATE(x, x_{near}, x_{new}, u_{new}) then if collision free calculate

T.add_vertex(x_{new})

T.add_edge(x_{near}, x_{new}, u_{new})

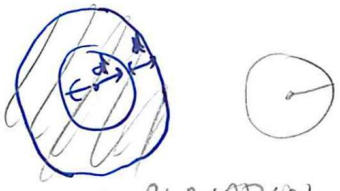
if x_{new} = x then

Return Reached

else

Return Advanced

Return Trapped.



EXPLANATION :

(1) Explanation for inflating obstacle already with

(2) We then spawn a random state x_{rand} & check if it is belonging to space X_{obs}.

(3) We reject x_{rand} if it $\in X_{obs}$, Now, we identify the closest node x_{near} in the current tree.

(4) On the line joining x_{near} & x_{rand}, we consider the value V & w for which we move towards x_{rand}.

(5) We repeat step 4 till it reaches x_{rand}

(6) Repeat steps 2, 3, 4 again till we reach x_{goal}.

(b) (i) Basic RRT can get trapped in the tree & could never reach the goal

(ii) Inefficient for one-time queries

(iii) So, we bias the RRT towards goal by sampling x_{rand} as x_{goal} based on ~~coin~~ coin toss which results in faster reach.

Goal bound:

(i) No suitable metric to decide closeness b/w 2 states

(ii) Difficult to reach final state exactly through discrete search.

Q7) Consider a drone that starts from a location (x_0, y_0, z_0) which strives to reach (x_g, y_g, z_g). (a) Formulate a quadratic goal reaching cost function that computes a set of controls to reach the goal. **(1 points)** (b) What are the constraints you may want to add to the above cost function. Do not worry about the obstacle avoidance constraint. Write down all other constraints that you feel are essential. Explain these constraints. **(1 points)** (c) Consider an obstacle of radius R, centered at (x_{ob}, y_{ob}, z_{ob}). Formulate an obstacle avoidance constraint that enables optimizing over a sequence of controls to eventually reach the goal. **(1 points)** (d) Linearize the obstacle avoidance constraint. **(2 points)**

(a) Minimize. $(x_n - x_g)^2 + (y_n - y_g)^2 + (z_n - z_g)^2$

(b) We have to add kinematic constraints:

$$\left.\begin{aligned} x_n &= x_{n-1} + \dot{x}_{n-1}\Delta t \\ x_{n-1} &= x_{n-2} + \dot{x}_{n-2}\Delta t \\ &\vdots \\ x_1 &= x_0 + \dot{x}_0 \Delta t \end{aligned}\right\} \quad x_n = x_0 + \sum_{i=1}^{n-1} \dot{x}_i \Delta t \quad \rightarrow (1)$$

similarly for y_n & z_n.

Now, the motion model of drone could be different, & so then constants will be different depending on that.

(c) Obstacle constraint: $(x_n - x_{ob})^2 + (y_n - y_{ob})^2 + (z_n - z_{ob})^2 > R^2$

$$-\left((x_n - x_{ob})^2 + (y_n - y_{ob})^2 + (z_n - z_{ob})^2 - R^2\right) < 0. \quad \rightarrow (2)$$

Let us take 2 waypoints for convenience sake:

From (1) in (2),

(d) $\left(x_0 + (\dot{x}_0 + \dot{x}_1)\Delta t - x_{ob}\right)^2 + \left(y_0 + (y_0 + \dot{y}_1)\Delta t - y_{ob}\right)^2 + \left(z_0 + (z_0 + \dot{z}_1)\Delta t - z_{ob}\right)^2$

(Ignore R^2 for now)

Linearize: $f(\dot{x}_0, \dot{x}_1) = f(\dot{x}_0^*, \dot{x}_1^*) + \left[\frac{\partial f}{\partial \dot{x}_0} \; \frac{\partial f}{\partial \dot{x}_1}\right]\Big|_{\dot{x}_0^*, \dot{x}_1^*} \begin{bmatrix} \dot{x}_0 - \dot{x}_0^* \\ \dot{x}_1 - \dot{x}_1^* \end{bmatrix}$

Just take 1^{st} for:

$$2\Delta t (x_0 - x_{ob}) + 2\Delta t^2 \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_0 \\ \dot{x}_1 \end{bmatrix}$$

Writing in matrix form:

$$A(x - x^a) + B(y - y^a) + C(z - z^a) - R^2 < 0$$

$$\Rightarrow (2A_1 x + 2q_1)^T \Big|_{x \to x^a} \to A_1 \to \Delta t^2 \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

$$q_1 \to (x_a - x_b)\, \Delta t \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

Similarly for B & C.

Extra space

Extra space

Extra space