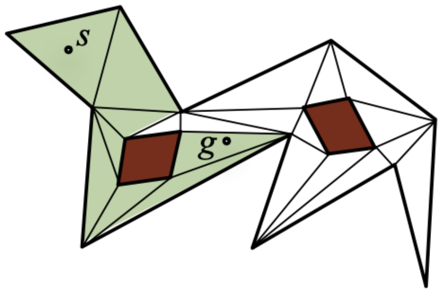
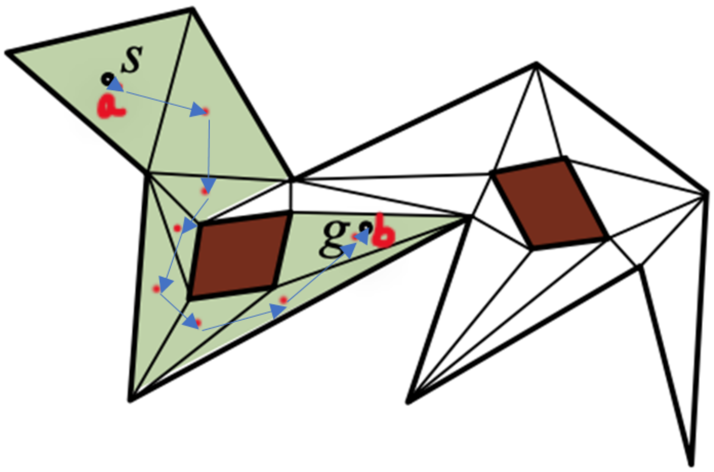
1. Point robot in a 2-D polygonal environment with obstacles in red. Green denotes a collision-free channel in the free space, white denotes all triangles in the free space. Start position *s*, goal position *g*.



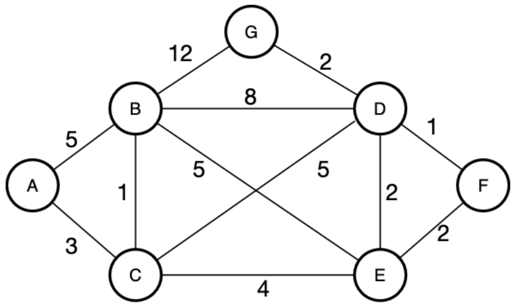
   a. Search Graph *G* = (Nodes *N*, Edges *E*). The graph is undirected.
   Nodes *N* = The centroid of each triangle.
   Edges *E* = A line segment connecting the centroids (nodes *N)* of 2 adjacent triangles in the free space. The weight of each edge can be taken to be a constant such as equal to 1. No edges are formed between 2 triangles when they are separated by an obstacle as they are not adjacent anymore.
   To find the channel: <clarify that triangles encapsulating nodes would become channel>

   b. **Input**: Channel σ, a sequence of adjacent triangles that contains a collision-free path connecting a start position *s* and a goal position *g*.
   Output: A collision-free path from *s* to *g*.
   Algorithm:

      i. Start from *s* (The 1st triangle in the σ contains *s*).
      Path = Φ
      ii. Connect *s* to the centroid of the triangle containing *s* (denoted by *a*) using a line segment *s'*.
      iii. Append *s'* to *Path*.
      iv. Loop over σ. For each triangle *T* in σ:
         1. *Source* = centroid of *T*
         2. If there exists an edge *E* from *Source* to Node $n \in N$ in *G* and the triangle *T'* containing *n* is not visited:
            a. Append *E* to *Path*.
            b. Mark *T* as a visited triangle.
      v. Connect *Source* to *g* using a line segment *g'* (since we have reached the last triangle in σ which contains the target *g*).
      vi. Append *g'* to *Path*.
      vii. Return *Path*.



2. Shortest paths (length in square brackets) to node A from -
   B: B → C → A [4]
   C: C → A [3]
   D: D → C → A [8]
   E: E → C → A [7]
   F: F → E → C → A;
   F → D → C → A [9]
   G: G → D → C → A [10]

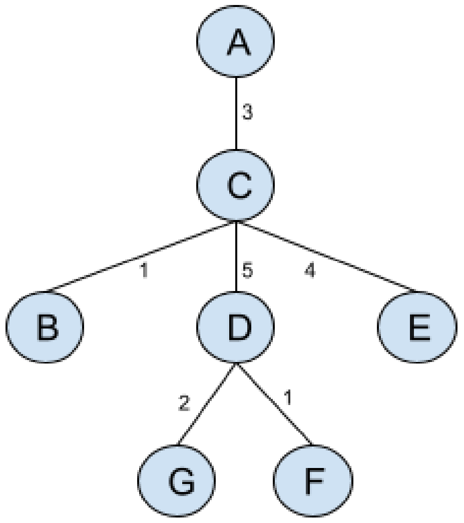


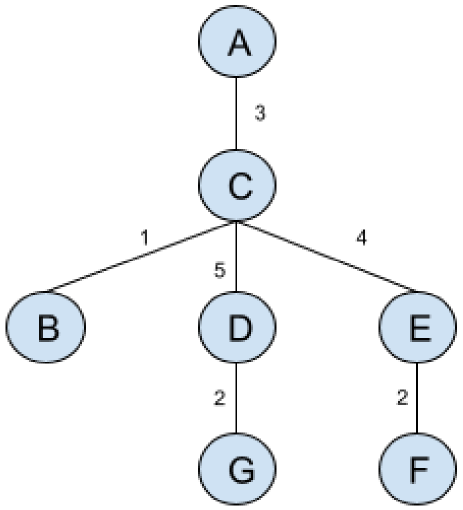
   a. Backward DP:

	A	**B**	**C**	**D**	**E**	**F**	**G**
$\mathbf{V_0}$	0	∞	∞	∞	∞	∞	∞
$\mathbf{V_1}$	0	5	3	∞	∞	∞	∞
$\mathbf{V_2}$	0	4	3	8	7	∞	17

	A	B	C	D	E	F	G
V_0	0	∞	∞	∞	∞	∞	∞
V_1	0	5	3	∞	∞	∞	∞
V*	0	4	3	8	7	9	10

b. Shortest path tree with root A (goal):





---

3. M=R is a 3×3 orthonormal matrix with determinant +1.

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

a. Constraints on R:
   i. $r_{11}^2 + r_{21}^2 + r_{31}^2 = 1$
   ii. $r_{12}^2 + r_{22}^2 + r_{32}^2 = 1$
   iii. $r_{13}^2 + r_{23}^2 + r_{33}^2 = 1$
   I.e., each column is a unit vector
   iv. $r_{11}\, r_{12} + r_{21}\, r_{22} + r_{31}\, r_{32} = 0$
   v. $r_{11}\, r_{13} + r_{21}\, r_{23} + r_{31}\, r_{33} = 0$
   vi. $r_{13}\, r_{12} + r_{23}\, r_{22} + r_{33}\, r_{32} = 0$
   I.e., the columns and rows are orthogonal to each other

b. det(M) = +1 doesn't reduce the degree of freedom of M. The above 6 constraints are independent constraints on the matrix entries. However, det(M) = +1 is an implicit constraint induced by the property of the resulting matrix becoming orthonormal (after applying the 6 constraints).

---

4. Configuration space dimensions.
   a. **2.** Since the base is fixed, there is no translation. The 2 revolute joints need 2 angles $\in [0, 2\pi)$ to parameterize their positions in the plane. $S^1 * S^1$.

   b. **6.** It takes 3 parameters for 1 freely translating and rotating robot in the 2D plane. 2 (for translation) + 1 (for rotation). Since there are 2 such robots and they are independent of each other, DOF is 3*2 = 6.

   c. **18.** It takes 6 angles to parameterize 1 revolute joint. Also, since the UAV must be moving (translating and rotating) in the 3D space, 3 points will parameterize its reference position and 3 points will parameterize its rotation. 6*2+3+3.

---

5. Infinite line *l* that translates and rotates freely in 3-D space.
   a. Parameterizations:
      i. **With angles:** Using the axis-angle specification, we have $(u, n_x, n_y, n_z, \theta)$ where (n_x, n_y, n_z) is a unit vector representing the axis of rotation, and θ is the angle of rotation about the axis.
      Constraint: $n_x^2 + n_y^2 + n_z^2 = 1$.
      ii. **Without angles:** Using the unit quaternion specification, we have (u, u_1, u_2, u_3, u_4) where (u_1, u_2, u_3, u_4) specify the direction of the line.
      Constraint: $u_1^2 + u_2^2 + u_3^2 + u_4^2 = 1$.
      iii. Since it is an infinite line, once we fix the angle of rotation, it extends indefinitely in that direction. Now, if the line translates, it will translate perpendicular to the direction in which it is extended. Hence, out of the 3 coordinates (x, y, z) in the 3-D plane, 2 of them are always fixed. Hence, we only need to parameterize 1 of them. This point is represented by u.

   b. The dimension of C is **4**. 5 parameters with 1 constraint in both cases.

   c. The dimension of the configuration space for *s* is **5** as we still need 3 parameters for translation and 3 for rotation with one constraint.
      i. Yes, we can use the 2 parameterizations in *(a)*. The modification would be to use 3 different parameters for representing a reference point in the line $\rightarrow (x, y, z)$ (instead of just 1) since the line segment is finite.

   ii. There would also be a constraint on the length of the line segment.

---

6. ** Done after question 11.

---

7. Sampling points uniformly
   a. A circle with centre *c* and radius *r*: $S = \{p \in R^2 \mid // p - c // = r\}$.
      i. Sample **$\theta \sim [0, 2\pi)$.**
      ii. Set p = (x, y) such that **$x = (r.\cos\theta + c), y = r.\sin\theta$.**
      iii. Since we know that arcs spanned by the same angles are sampled with equal probabilities, points on this circle will be uniformly sampled.

   b. A disc B_2 with centre c and radius r: $B_2 = \{p \in R^2 \mid // p - c // \leq r\}$.
      i. Set p = (x, y) such that **$x = (a.\cos\theta + c), y = a.\sin\theta$.**
      ii. Sample **$\theta \sim [0, 2\pi)$** and **$a \sim [0, r]$** separately.
      iii. This can be proved by solving the equation:
      $\rightarrow (x - c)^2 + y^2 \leq r^2$

   c. For rejection sampling, sample **$x \sim [c-r, c+r]$** and **$y \sim [-r, r]$** independently, and accept the sample p = (x, y) if $(x-c)^2 + y^2 \leq r^2$. I would choose rejection sampling as it is easier to implement than deciding the correct parameterization of the polar coordinates. For a much more complex geometry, absolute parameterization and/or sampling the sophisticated probability distribution of the geometry would become extremely tough. Rejection sampling is also faster to implement.

   d. Rejection rate:
      i. In rejection sampling, if we consider sampling from $p(x) = N(0, \sigma_p^2 I)$ using a proposal $q(x) = N(0, \sigma_q^2 I)$, the acceptance rate is 1/M, where:
         1. $\sigma_q >= \sigma_q$ in order to be an upper bound.
         2. $M = (\sigma_q/\sigma_p)^D$.
         3. D is the dimension. (Using D instead of N as specified in the question as N is being used for a normal distribution here).
      ii. Therefore, with increasing dimension D, the acceptance rate decreases exponentially.
      iii. Since the rejection rate is (1 - Acceptance Rate), in higher dimensions the majority of the samples would be rejected and rejection sampling would ultimately fail.

   iv. If the sphere is assumed to follow an approximate gaussian distribution (the above result is generalized to any geometry), the rejection rate exponentially increases with increasing D.

e. The Muller method states that the probability distribution of an N-D sphere only depends on its magnitude and not any direction. Thus it must be rotationally invariant (symmetric).
   i. Sample N normally distributed random variables.
   ii. Find the norm of the N random variables.
   iii. Define a scaling factor S where r is a random number → $S = r^{(1/N)}$.
   iv. Scale the N random variables by a factor of S and normalize using the nom.

---

8. Configuration space C is the unit square [0, 1] × [0, 1]
   a. If the algorithm calls LINK for every pair of roadmap nodes, the number of calls to LINK is equivalent to computing the number of ways of choosing 2 nodes from a set of n configurations, i.e., ${}^{n}C_2 = n*(n-1)/2$. For $n \to \infty$, the asymptotic upper bound complexity becomes $\mathbf{O(n^2)}$.

   b. The number of calls to LINK would be directly proportional to the number of milestones present within the threshold $t = 1/\sqrt{n}$ from a milestone. Since the milestones are uniformly distributed, we can assume to count all the milestones within a disk of radius t centred at a milestone.
      i. The density of the disk = Area of disk * Max. No. of points possible within the disk = $(\pi*t^2) * n = \pi/n * n = \pi$.
      ii. Since there are n milestones, the total number of calls to LINK = $\pi * n = \pi n$.
      iii. Thus, the asymptotic bound on the number of calls is constant → $\mathbf{O(n)}$.

---

9. Motion planning algorithm. Justify your choice by considering the configuration space dimensions, environment characteristics, and computational efficiency.
   a. **Online A***.
      i. The C-Space is very small ~3 DOF and the robot is expected to search for an open parking slot and then park itselfwhich makes the environment dynamic due to changing configurations of other cars in the parking lot.

   b. **Online A*** *or* **EST/RRT**.

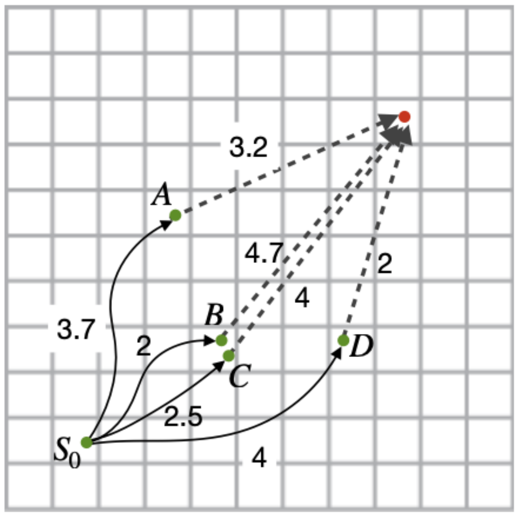
i. The C-Space is small since the robot is only translating in 3D to disinfect the area. The environment layout (area to be cleaned) is more or less fixed with the exception of a large number of people moving in the area due to the busy hours. This makes the environment dynamic.

ii. However, if the robot has a lot of moving parts used for cleaning, the C-Space becomes large. **In this case, the best algorithm is EST, or RRT**.

c. **Online EST, RRT.** The C-Space is extremely large due to a lot of modular parts. The environment, i.e., the task is dynamic as the robot needs to adjust itself on the fly in order to complete the task.

---

10. Hybrid A* algorithm.



a. The dimension of the grid is 2. This is justified as the heuristics use euclidean distance from an intermediate node n' to target t. This requires 2 points to compute in a 2D plane.

b. f = g + h, where g = cost-to-come; h = cost-to-go. Priority queue:
   i. D → 4 + 2 = 6
   ii. C → 2.5 + 4 = 6.5
   iii. B → 2 + 4.7 = 6.7
   iv. A → 3.7 + 3.2 = 6.9

---

11. Heuristic for A*.

a. **Given**: *$h_1(x)$, $h_2(x)$* are admissible heuristics.
To prove: *$h(x) = max\{h_1(x), h_2(x)\}$* is also admissible.
Proof:
   i. Consider the problem of going to an arbritrary target y from a source x.
   ii. By the defintion of admissibility, *$h_1(x)$* and *$h_2(x)$* never overestimate the cost of going from x → y over the true cost *$T(x)$*, i.e.,
   $h_1(x) <= T(x)$
   $h_2(x) <= T(x)$
   iii. Since, *$h(x) = max\{h_1(x), h_2(x)\}$,*
   $h(x) <= T(x)$, for the same goal-target x → y.
   iv. Therefore, *$h(x)$* never overestimates the the cost of going from x → y over the true cost *$T(x)$*.
   v. Hence, *$h(x)$* is admissible.

b. I will use h(x) since it will always choose the best possible heuristics at all times. Since admissible heuristics are a **pessimistic bound** on the true cost, a good heuristic is one which matches the true cost as much as possible (in order to give better approximations to the true cost). Therefore, the aim is to choose a heuristic which has the **minimum possible bound between its cost and the true cost**.
Proof:
   i. We know that,
   $h_1(x) <= T(x)$
   $h_2(x) <= T(x)$
   $h(x) = max\{h_1(x), h_2(x)\} <= T(x)$
   ii. Let the differences between the true cost and the heuristics be defined as:
   $a = T(x) - h_1(x), a >= 0$
   $b = T(x) - h_2(x), b >= 0$
   $c = T(x) - h(x), c >= 0$
   The goal is to choose a heuristic which has the minimum of {a, b, c}.
   iii. Therefore,
   $\rightarrow h(x) = \max\{T(x) - a, T(x) - b\}$
   $= T(x) + \max\{-a, -b\}$
   $= T(x) - \min\{a, b\}$
   $\rightarrow T(x) - c = T(x) - \min\{a, b\}$
   $c = \min\{a, b\}$
   I.e., out of the 3 given bounds between the true cost and the heuristic, h(x) has the lowest bound.
   iv. Therefore, its the best heuristic out of the 3.

6. Configuration space distance and workspace distance.

(a) **Given**:
$q = (\theta_1, \dots, \theta_n)$; $q' = (\theta_1', \dots, \theta_n')$
$d_c(q, q') = \max |\theta_i - \theta_i'|\ 1 \leq i \leq n$
Robot moves along the path $(1 - \lambda)q + \lambda q'$ for $0 \leq \lambda \leq 1$.

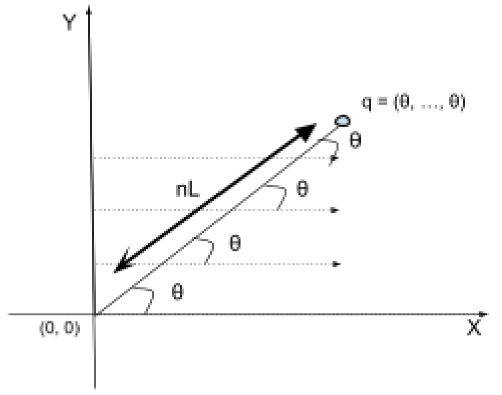
To prove: No point on the robot traces a path longer than $\alpha d_c(q, q')$ for $\alpha > 0$.

Proof:

1. Let the base of the robot be fixed at the origin (0, 0).

2. Considering the set of all the configurations (S) this robot has, the furthest point on the robot would be the extremity of the last link when all the links form a straight line, i.e., distance of extremity of the nth link from the origin = nL. **This is the maximum length the robot can achieve**.
Let this point be $q = (\theta_1, \dots, \theta_n) \in S$.

3. Since the individual links make the same angle from the base, $\theta_1 = \theta_2 = \dots = \theta_n = \theta$.



4. If we consider a circular region C of radius nL centered at the base of the robot, $S \in C$, i.e., the circumference of C is an upper bound to all the paths the robot can trace.

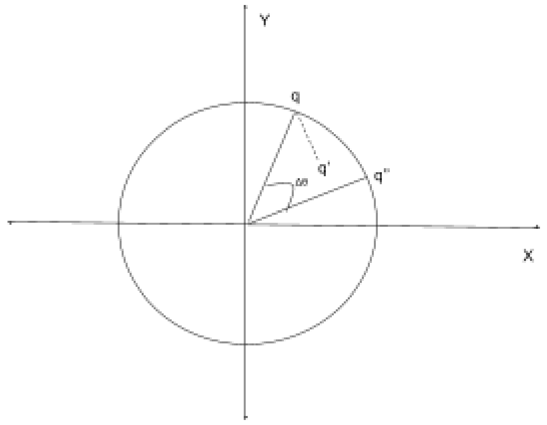
5. Let $d_w(q, q')$ = distance traced by robot in the workspace.

6. To show that no point on the robot traces a path longer than $\alpha d_c(q, q')$ for $\alpha > 0$, we prove that the furthest point q on the robot follows the relation $d_w(q, q') <= \alpha d_c(q, q')$ for $\alpha > 0$. **Since every other point traces a distance less than the distance traced by q (as C is an upper bound) , all points implicitly follow the relation.**

7. Since the robot moves from $q = (\theta , \dots , \theta)$ to a configuration $q' = (\theta_1' , \theta_2' , \dots , \theta_n')$ along the path $(1 - \lambda)q + \lambda q'$ for $0 \leq \lambda \leq 1$, q' has to lie either within or on the circle C.

8. If we consider the straight-line links configuration of the robot, and move only the 1st link with $\Delta\theta$ to move to another point q'' on the circle, the extremity of the last link will trace the arc of the circle C.
   $\rightarrow q'' = (\Delta\theta+\theta,\ \Delta\theta+\theta, \dots ,\ \Delta\theta+\theta)$.

9. Let this distance be $d_{arc}(q, q'') = nL * \Delta\theta$. $\rightarrow$ (Equation 1)



10. We know in R^n, $d_{arc}(q, q'') >= d_w(q, q')$. $\rightarrow$ (Equation 2)

11. Since $d_c(q, q'') = \max |\theta - \Delta\theta - \theta|\ 1 \leq i \leq n$, $d_c(q, q'') = \Delta\theta$. $\rightarrow$ (Equation 3).

12. Combining Equation 1, 2 and 3,
    $d_w(q, q') <= d_{arc}(q, q'') = nL * \Delta\theta = nL * d_c(q, q'')$.

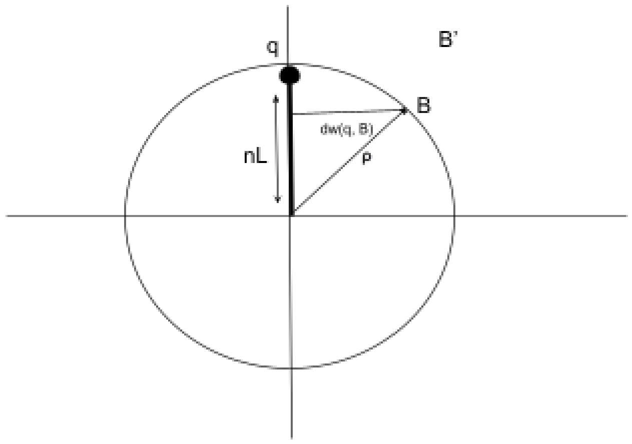
13. Since, in the extreme/ arbitrary case, q' = q'' (as q'' lies on C),
    $d_w(q, q') <= nL * d_c(q, q') = \alpha * d_c(q, q')$

14. **Therefore, $\alpha = nL$.**

(b) **Given:** $d_w(q, B)$ denotes the minimum distance between the robot placed at configuration q and a workspace obstacle B.

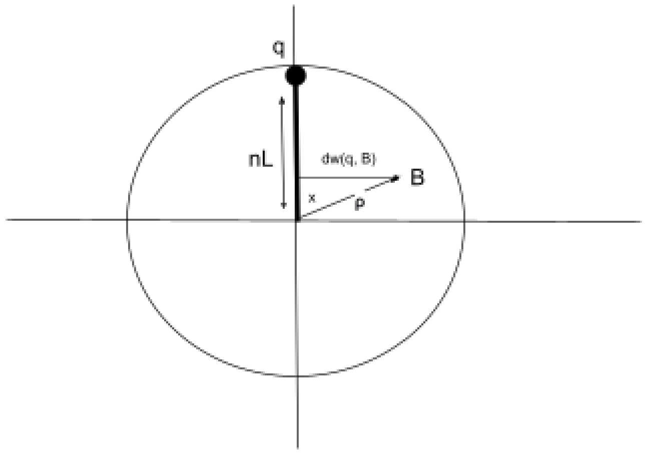
To find: Radius ρ of the neighborhood $N(q) = \{q' \mid d_c(q,q') \leq \rho\}$ in which the robot is guaranteed to move freely without colliding with B.

Case 1: B lies on or outside the circle C.



- In this case, the maximum area the robot can span is the circle C with radius **$\rho = nL = \alpha$**.

Case 2: B lies inside the circle C.



In this case,
$x^2 + d_w(q, B)^2 = \rho^2$
From (a), we know $x <= nL = \alpha$.
$\rightarrow x^2 + d_w(q, B)^2 <= \alpha^2 + d_w(q, B)^2$
$\rightarrow \rho <= \sqrt{(\alpha^2 + d_w(q, B)^2)}$

Therefore, combining Case 1 and Case 2, $\rho = \min\{ \sqrt{(\alpha^2 + d_w(q, B)^2)}, \alpha \}$.