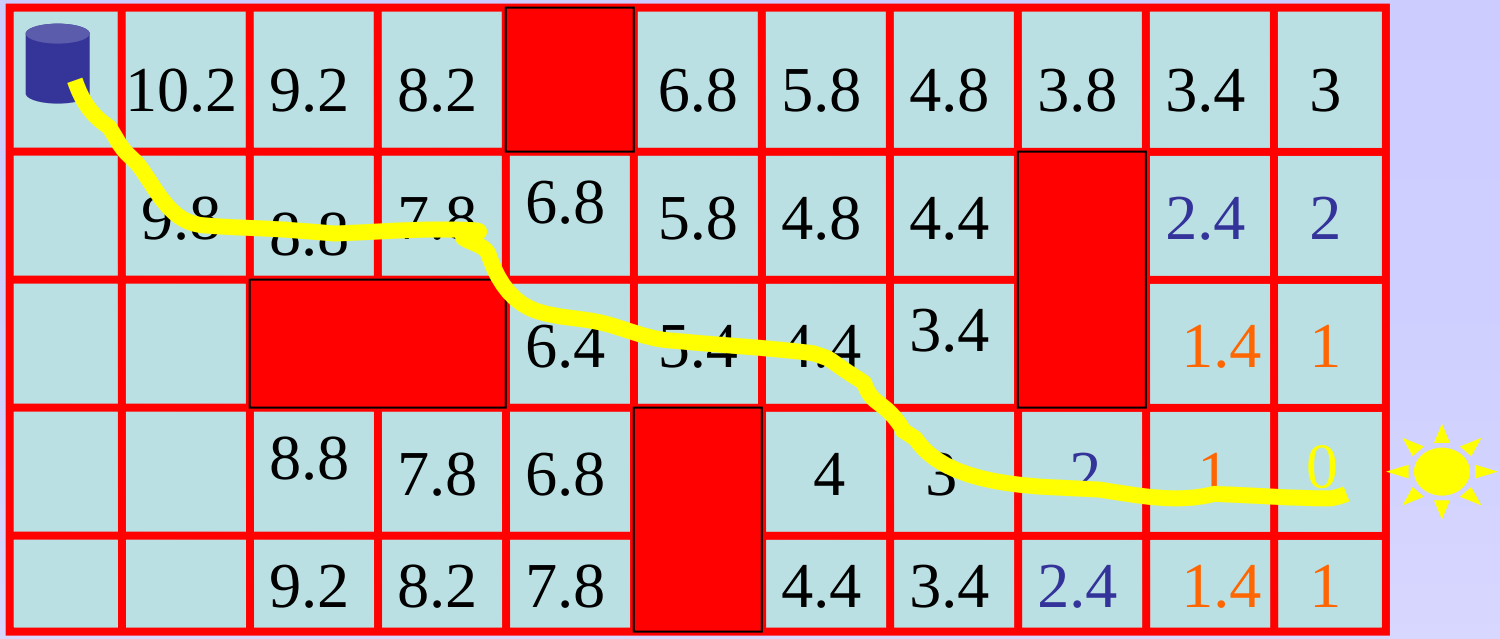
INSTITUTO NACIONAL DE ASTROFÍSICA, ÓPTICA Y ELECTRÓNICA

Introducción a la Robótica

L. Enrique Sucar

INAOE



Sesión 7: Planeación de trayectorias

Introducción a la Robótica

L. Enrique Sucar

Contendio

- Introducción
- Técnicas clásicas
  - Búsqueda en grafos
  - Programación dinámica
  - Diagramas de Voronoi
  - Grafos de visibilidad
  - Campos potenciales

Planeación de Trayectorias

- Determinar una trayectoria en el espacio de configuraciones, entre una configuración inicial (inicio) y una configuración final (meta), de forma que el robot no colisione con los obstáculos y cumpla con las restricciones cinemáticas del robot

Planeación de Trayectorias

- Consideraciones adicionales:
  - Trayectoria de distancia (tiempo o costo) mínimo
  - Ambientes semidesconocidos
  - Ambientes dinámicos
  - Restricciones adicionales
  - Eficiencia (algoritmos *anytime*)

Plan

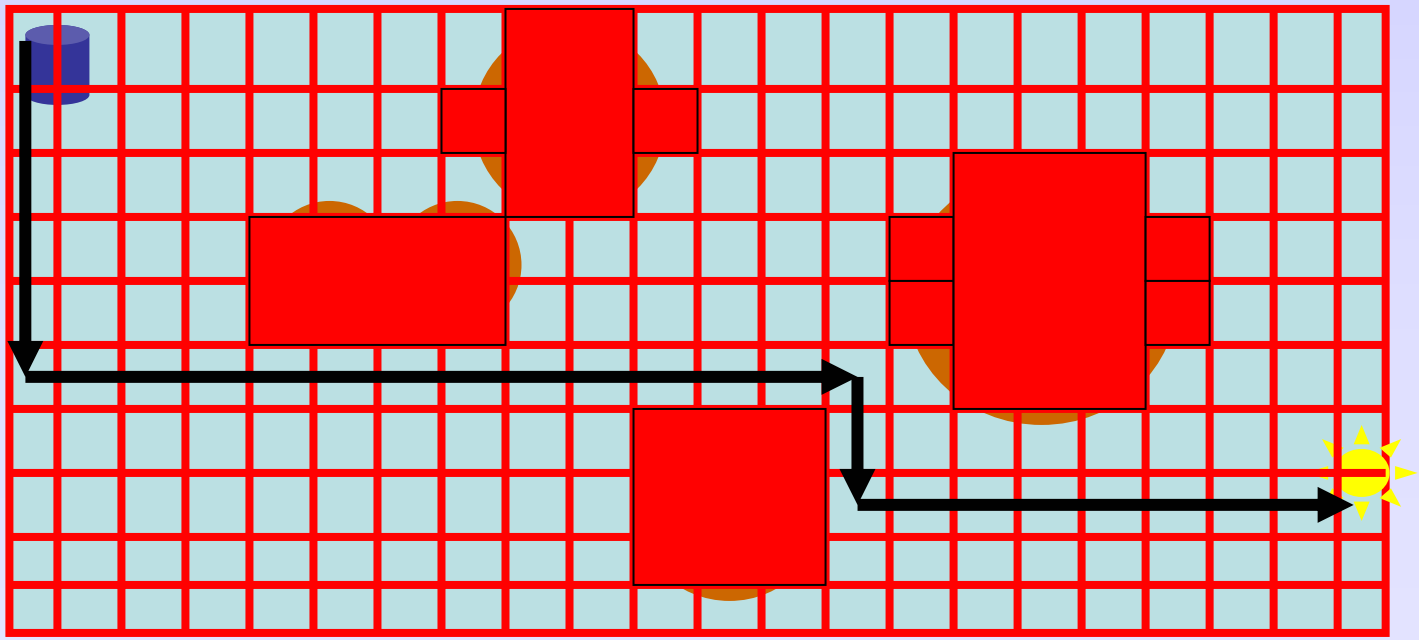
- Un plan es un conjunto de acciones (operadores) que permiten a un agente (robot) ir de un estado inicial a un estado final o meta
- Los elementos básicos para hacer un plan son:
  - Estados (p. ej. Posición del robot), incluyendo el estado incial y el estado meta
  - Operadores: acciones que llevan de un estado a otro, Si → Sj

Ejemplo de Plan

- Considerando el mapa de rejilla:
- Estados:
  - posición X,Y en el mapa
  - Estado inicial: 0,0
  - Estado meta (luz): Xm, Ym
- Acciones:
  - Movimiento a alguna de las celdas vecinas
  - X+1, Y+1, X-1, Y-1

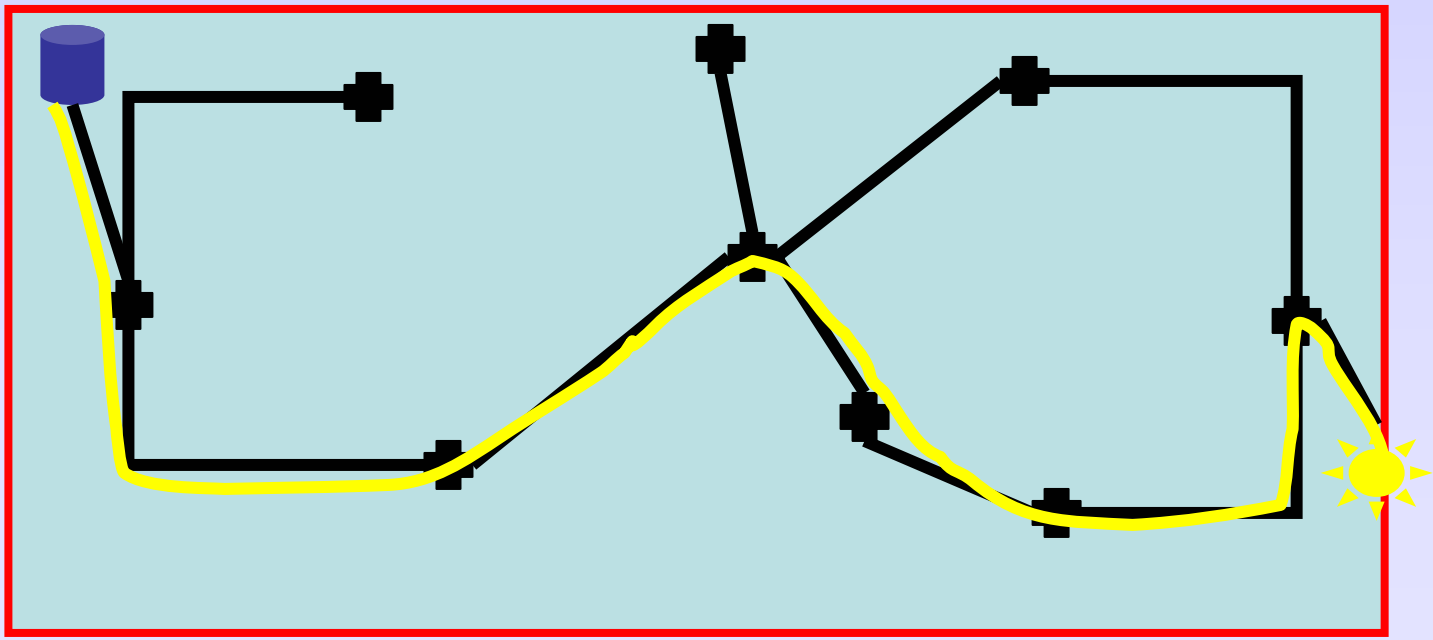
Ejemplo de Plan

- Plan: buscar una serie de acciones básicas que lleven al robot de la posición inicial a la meta



Ejemplo: plan en el mapa topológico

- Plan = búsqueda de una trayectoria en el *grafo,* del nodo inicial al meta



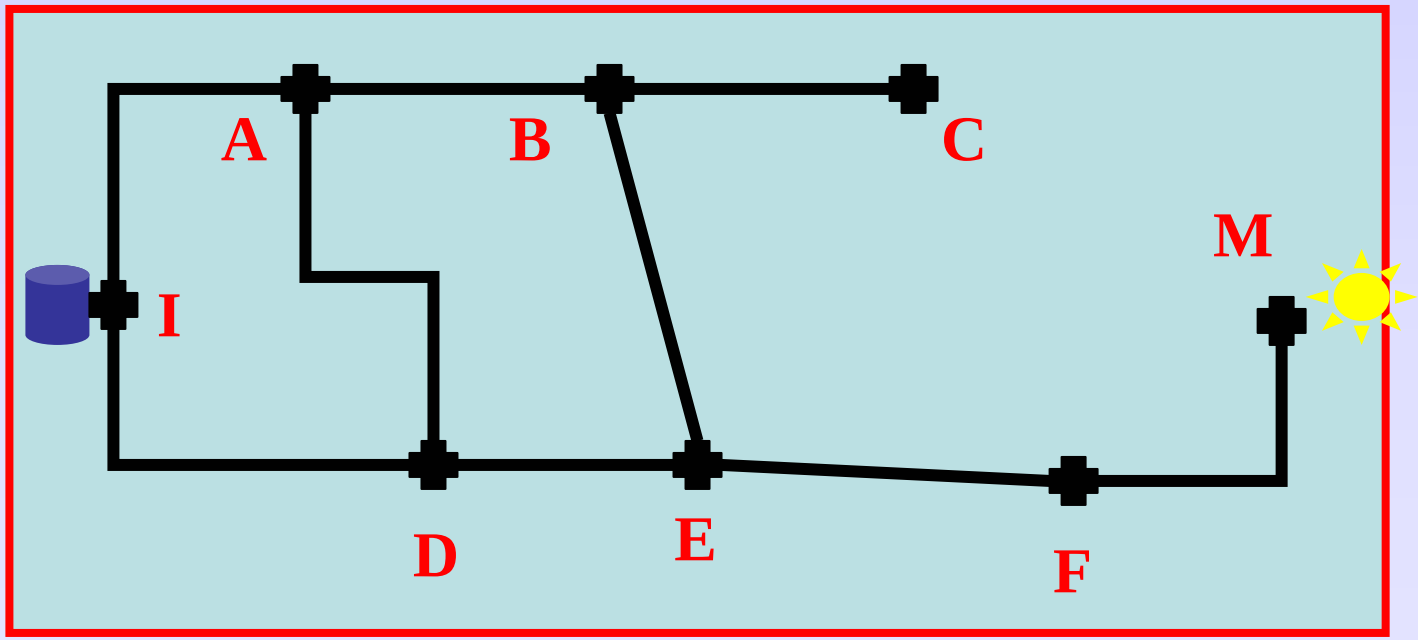
Consideraciones iniciales

- Un robot rígido usualmente modelado como un “punto”
- Un ambiente estático con obstáculos conocidos
- Un robot capaz de navegar en segmentos de línea recta y de mantener su localización

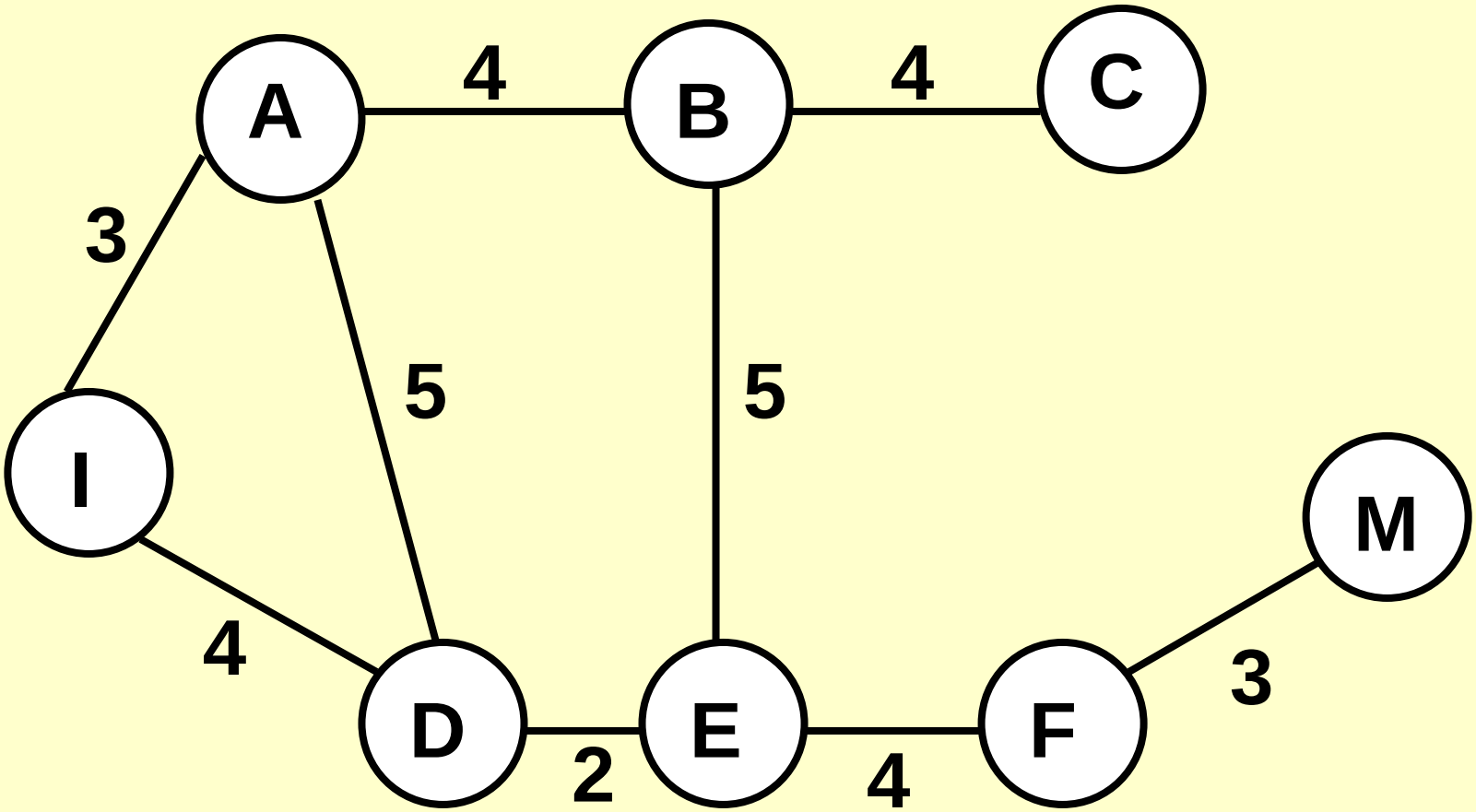
Técnicas Clásicas

- Espacios discretos
  - Búsqueda en grafos
  - Programación dinámica
  - Diagramas de Voronoi
  - Grafos de visibilidad
- Espacios continuos
  - Campos potenciales

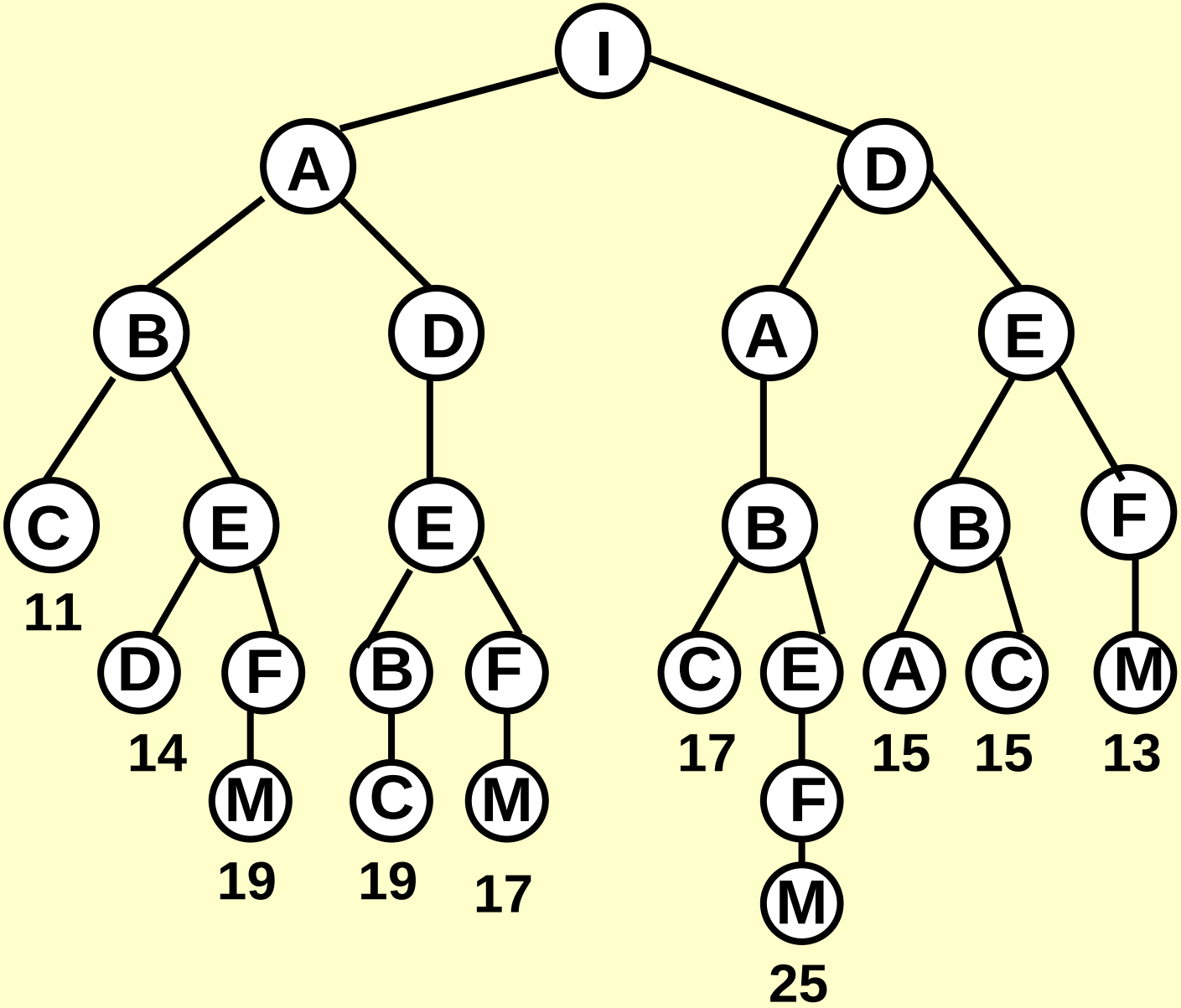
Ejemplo – técnicas de búsqueda



Ejemplo: ESTADOS Y DISTANCIAS



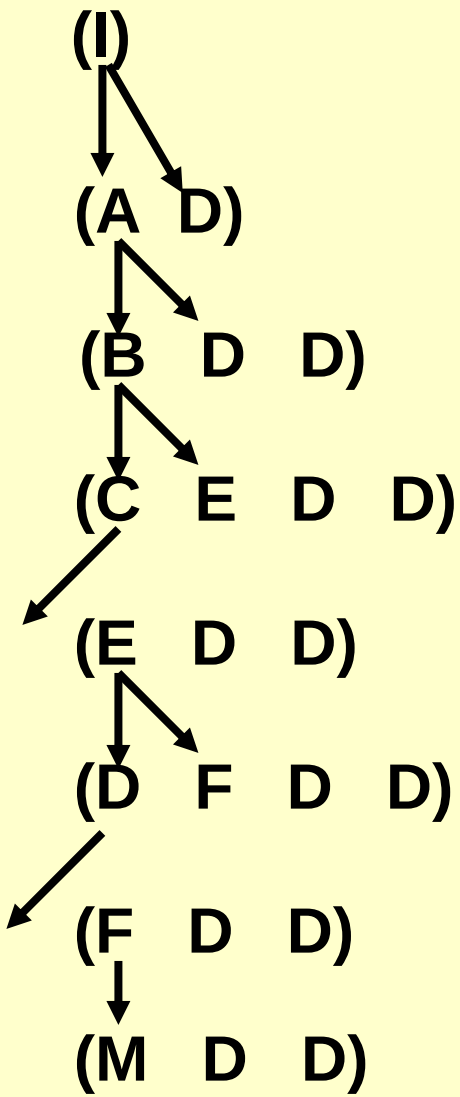
ÁRBOL DE BÚSQUEDA

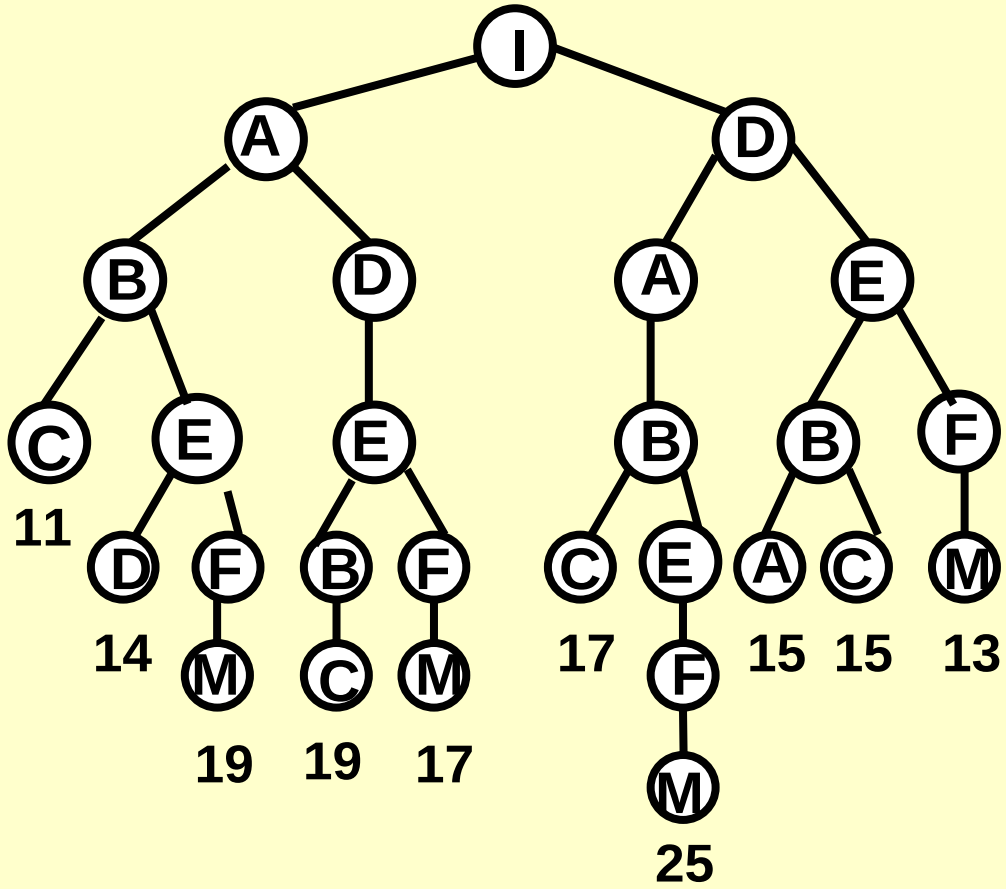


Depth first - backtracking (LIFO)

Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda este vacía o se alcance la meta
si el primer elemento es la meta
entonces acaba
si no elimina el primer elemento y
añade sus sucesores al *frente* de la agenda

DEPTH-FIRST SEARCH



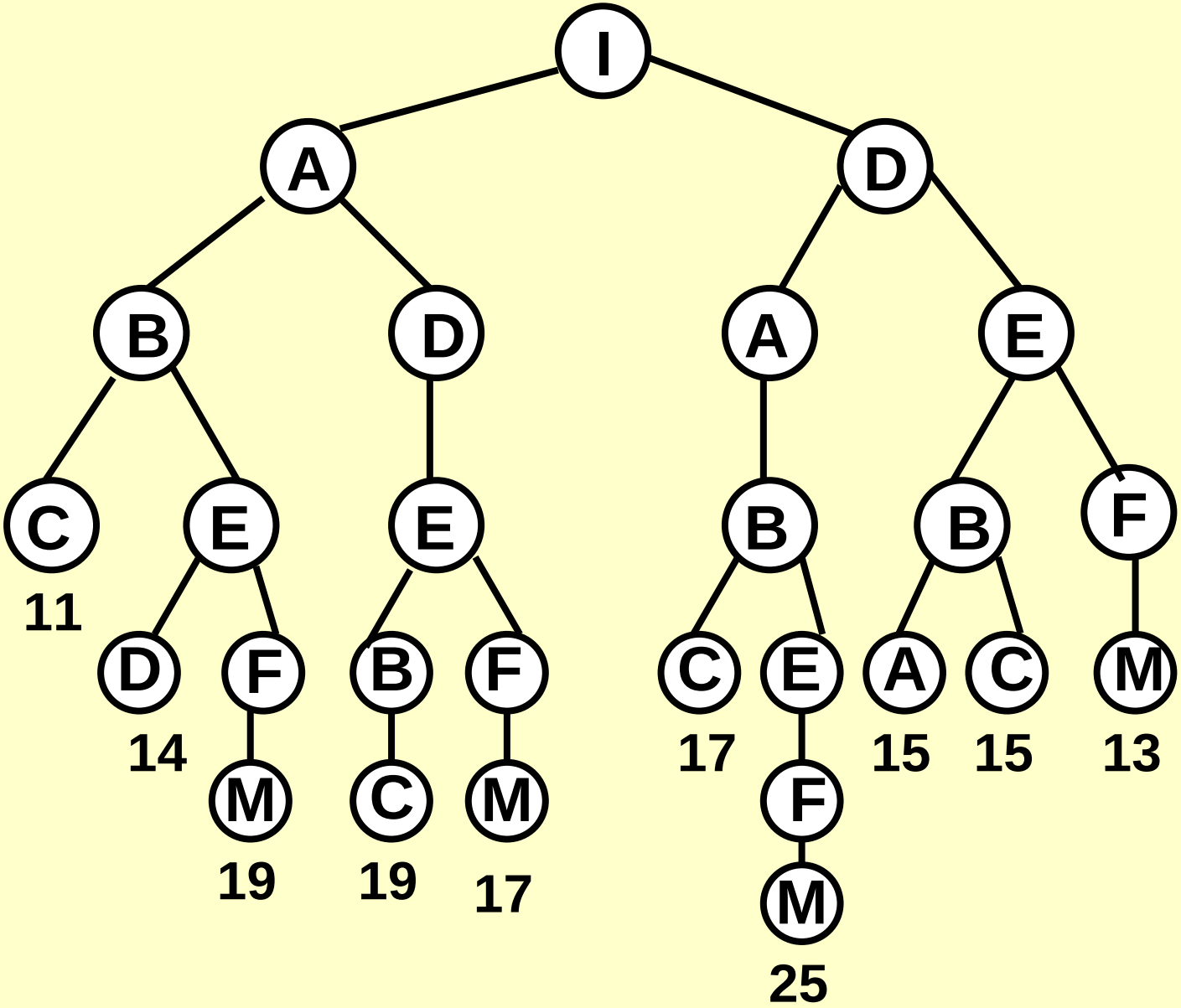


Breadth first

Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda este vacía o se alcance la meta
si el primer elemento es la meta
entonces acaba
si no elimina el primer elemento y
añade sus sucesores al *final* de la agenda

Problemas: árboles con arborescencia muy grande

ÁRBOL DE BÚSQUEDA



Algoritmos con Información

Hill-Climbing

Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda este vacía o se alcance la meta
si el primer elemento es la meta
entonces acaba
si no elimina el primer elemento y añade sus sucesores a la agenda
ordena todos los elementos de la agenda
selecciona el mejor y *elimina el resto*

BÚSQUEDA HILL-CLIMBING

Hill climbing

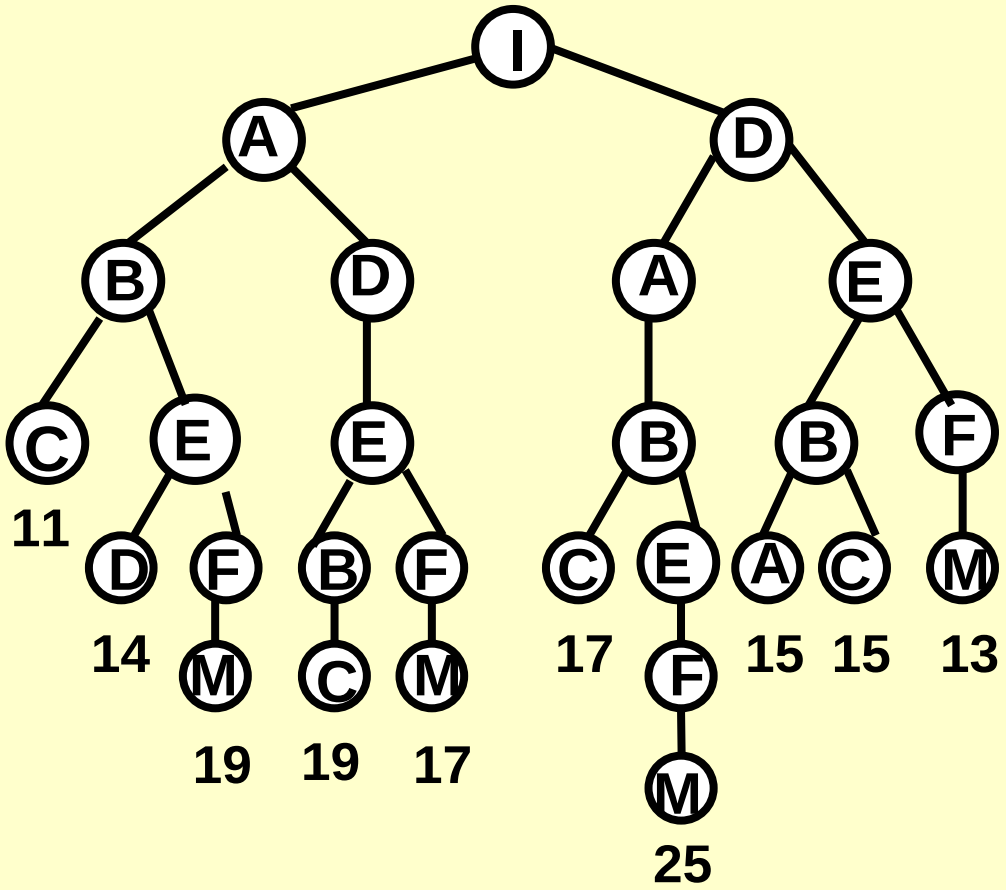
Heurística: ve a la ciudad más cercana

(I)

(A ~~D~~)

(~~D~~ B)

(~~E~~ C)



Beam search

Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda este vacía o se alcance la meta
si el primer elemento es la meta
entonces acaba
si no elimina el primer elemento y añade sus sucesores a la agenda
ordena todos los elementos de la agenda y selecciona los *N* mejores (los demás eliminalos)

A*: utiliza dos medidas

Idea: usar estimaciones de los costos/distancias que faltan junto con los costos/distancias acumuladas

$$\text{estim(total)} = \text{costo(camino recorrido)} + \text{estim(camino que falta)}$$

Las estimaciones no son perfectas, por lo que se usan sub-estimaciones

subestim(total) = costo(camino recorrido) + subestim(camino que falta)

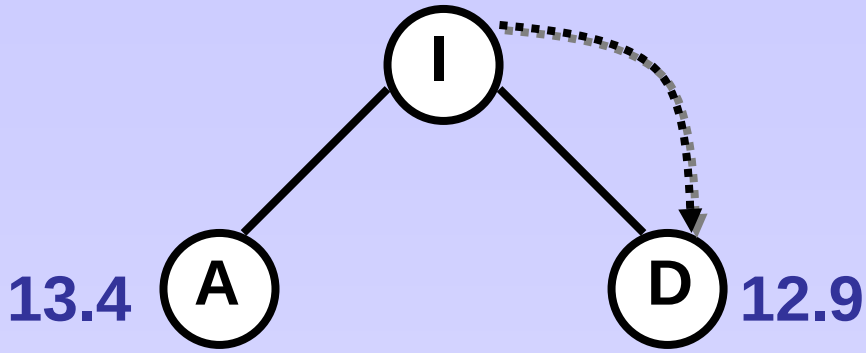
De nuevo expande hasta que los demás tengan sub-estimaciones más grandes (v.g., subestimaciones de distancias entre ciudades pueden ser líneas rectas)

Crea una agenda de un elemento (el nodo raíz)
hasta que la agenda este vacía o se alcance la meta y los demás nodos sean de costos mayores o iguales a la meta
si el primer elemento es la meta y los demás nodos son de menor o igual costo a la meta
entonces acaba
si no elimina el primer elemento y añade sus sucesores a la agenda
ordena todos los elementos de la agenda de acuerdo al costo acumulado más las subestimaciones de los que falta

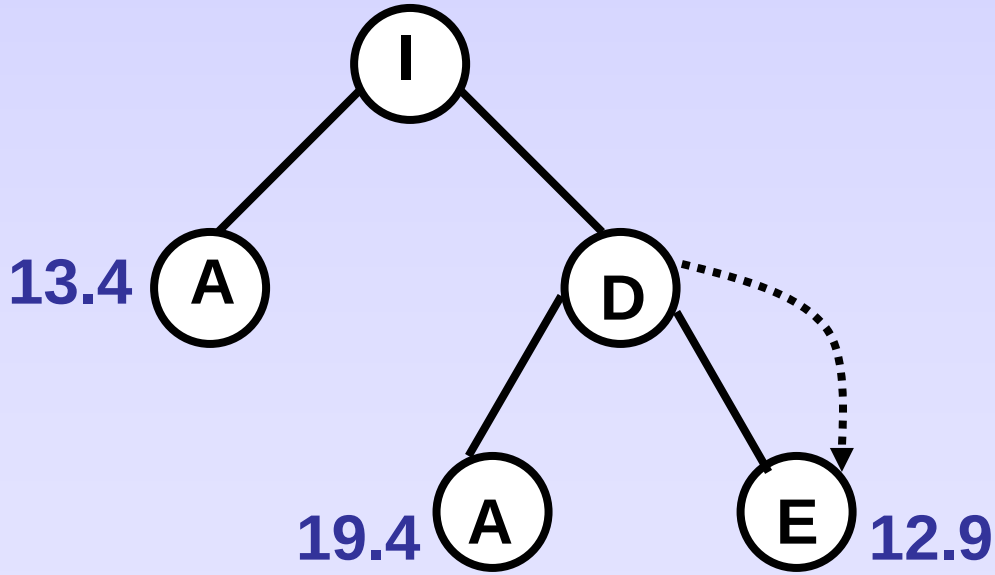
EJEMPLO DE BÚSQUEDA A*

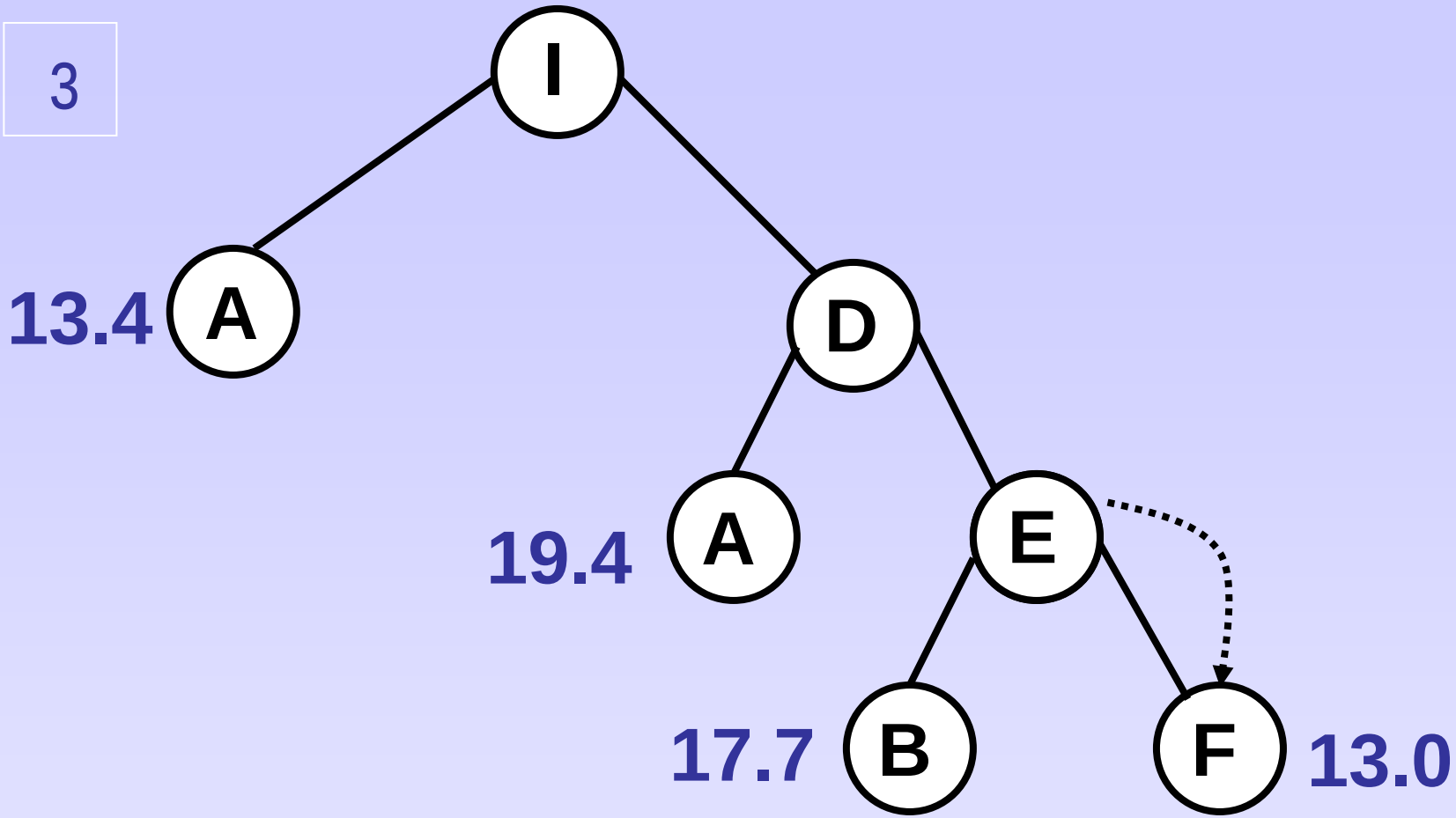
A*

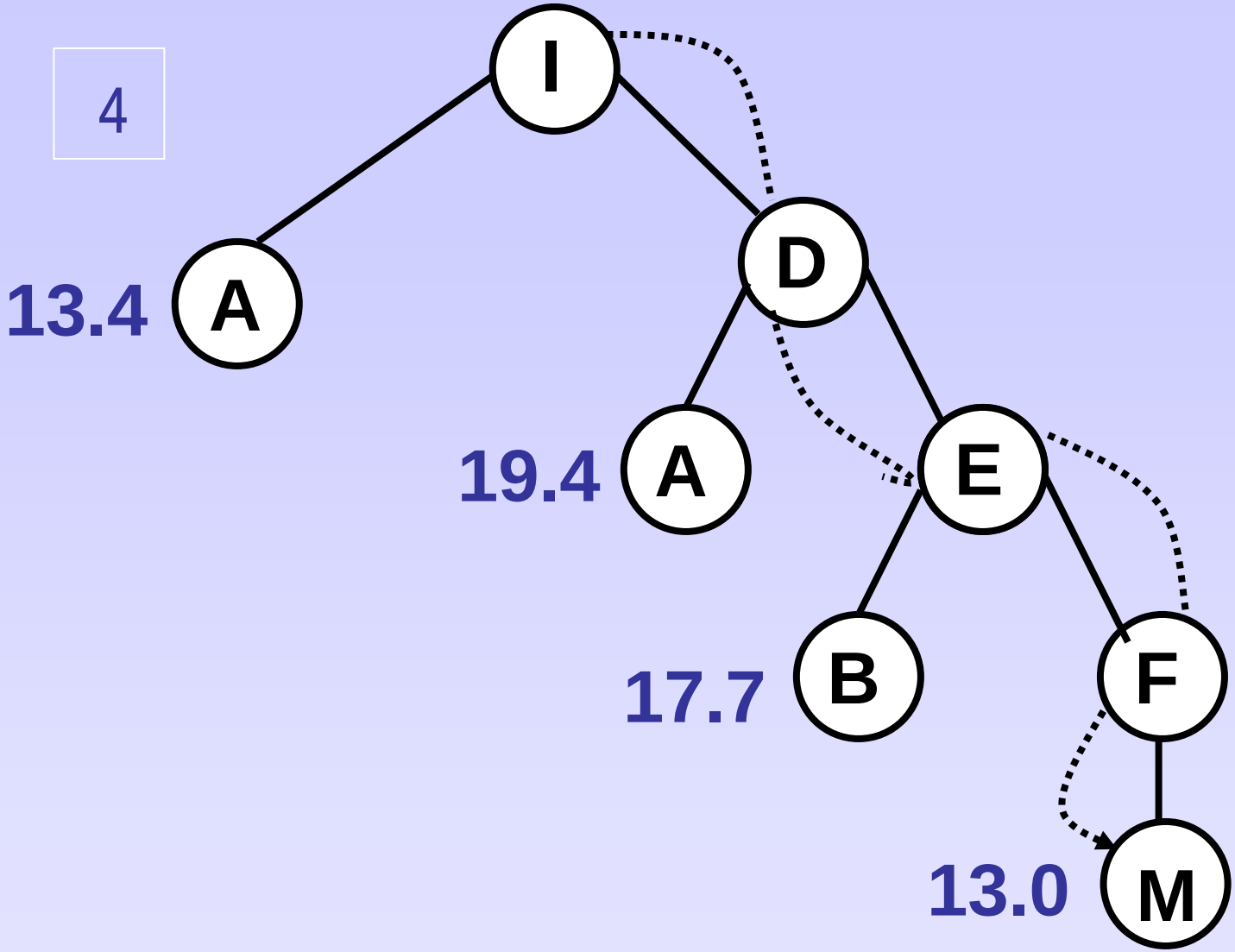
1



2




3
I
13.4
A
D
19.4
A
E
17.7
B
F
13.0


4
I
13.4
A
D
19.4
A
E
17.7
B
F
13.0
M

Programación dinámica

- Procedimiento iterativo (recursivo) para evaluar el costo de la trayectoria mínima de cualquier punto a la meta
- Se considera un ambiente discreto y un costo de moverse de un sitio (celda) a otro
- El algoritmo básico se conoce como “iteración de valor”

Algoritmo de *Programación Dinámica*

V(x,y) : Costo acumulado de viaje a la celda más cercana, si el robot está en la celda (x,y)

Inicio:

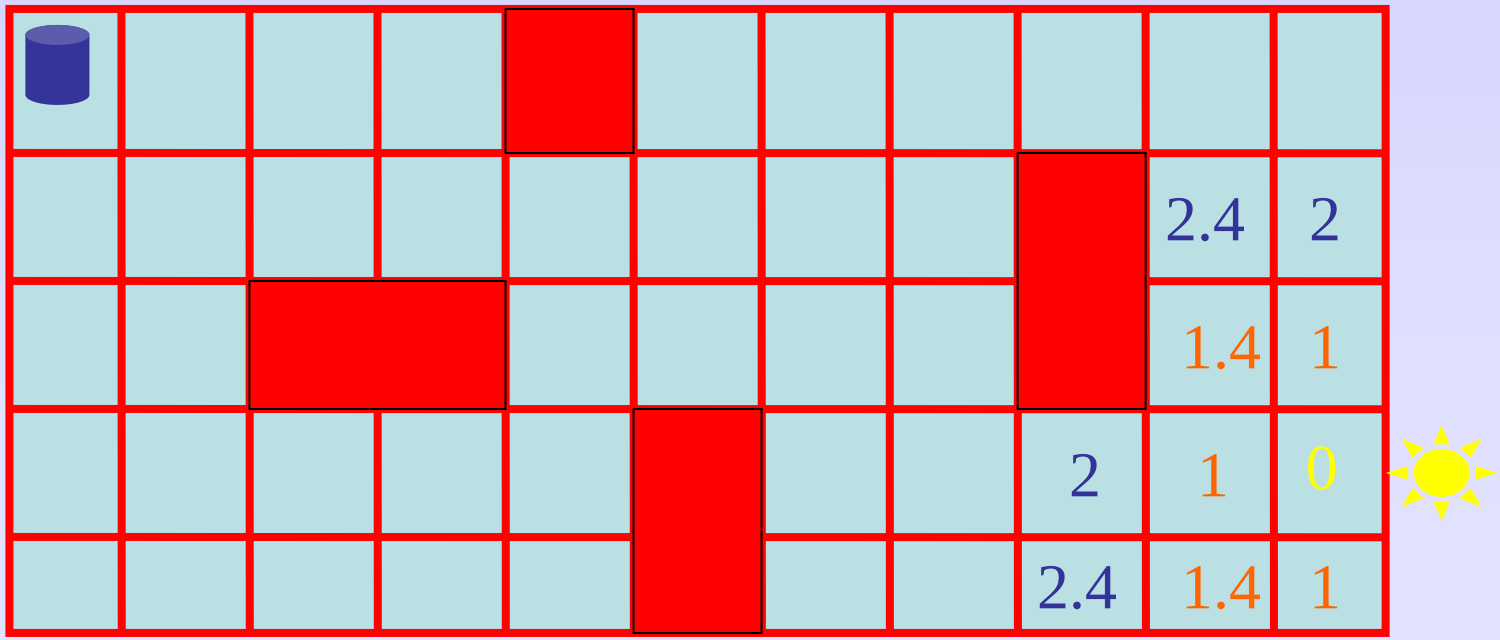
$V_i(x,y) = 0$ si la celda (x,y) es la meta

$= \text{infinito}$ demás celdas

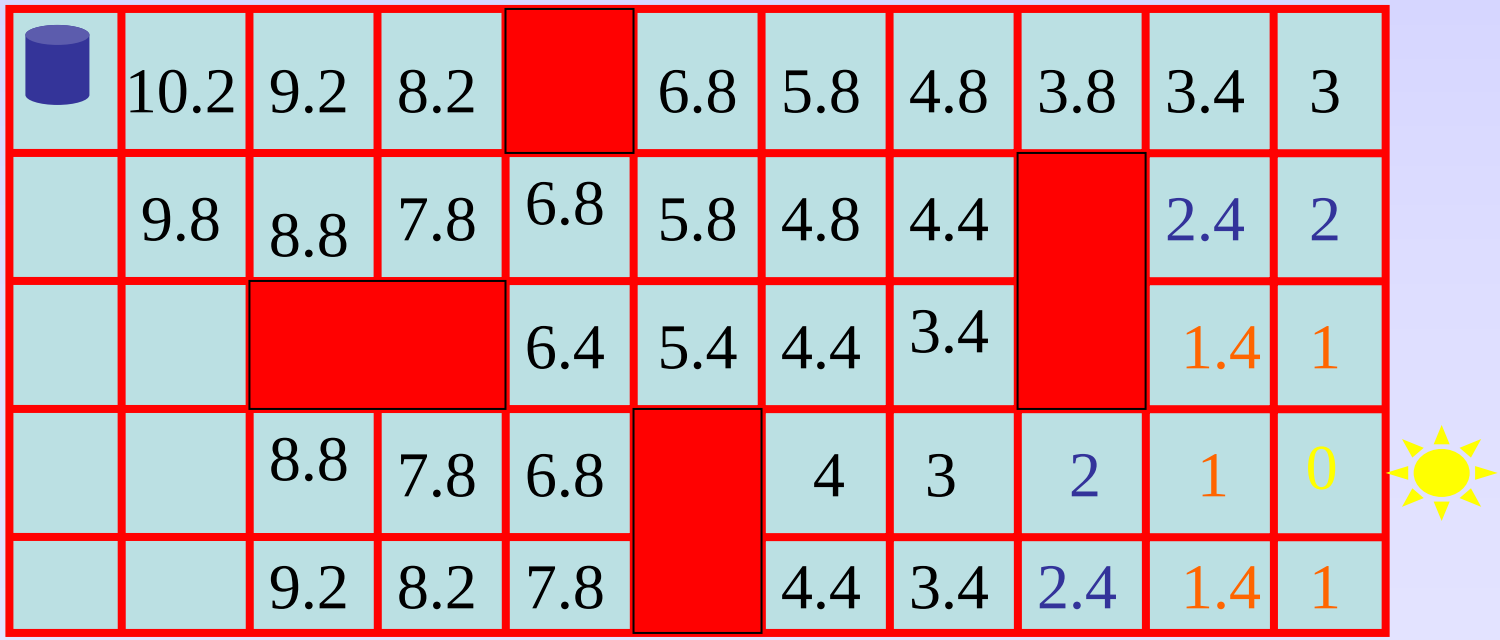
Regla de actualización:

$$V_i(x,y) = \min_{m(dx,dy) \text{ en Mov}} \{V_j(x+dx,y+dy) \text{ (Sig. celda)} + C_{mov}((x,y),(dx,dy))\} \quad \text{(avanzar)}$$

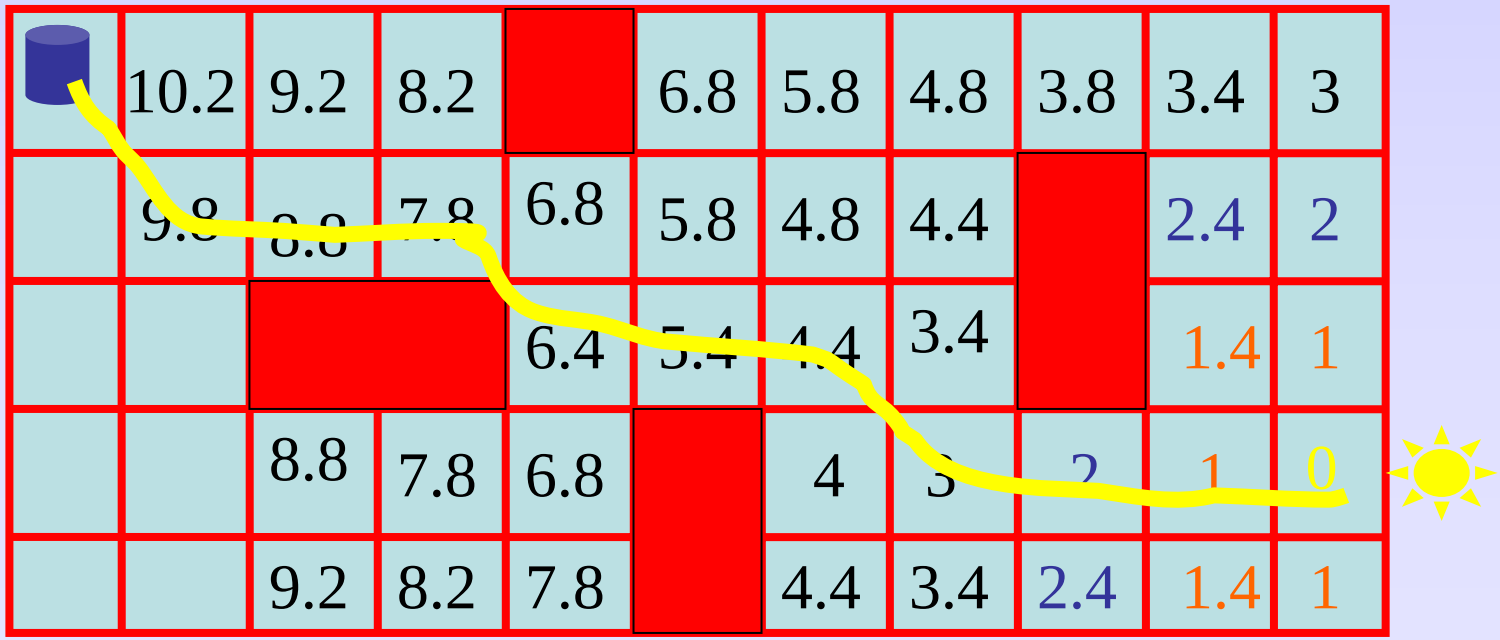
Ejemplo – programación dinámica



Ejemplo – programación dinámica

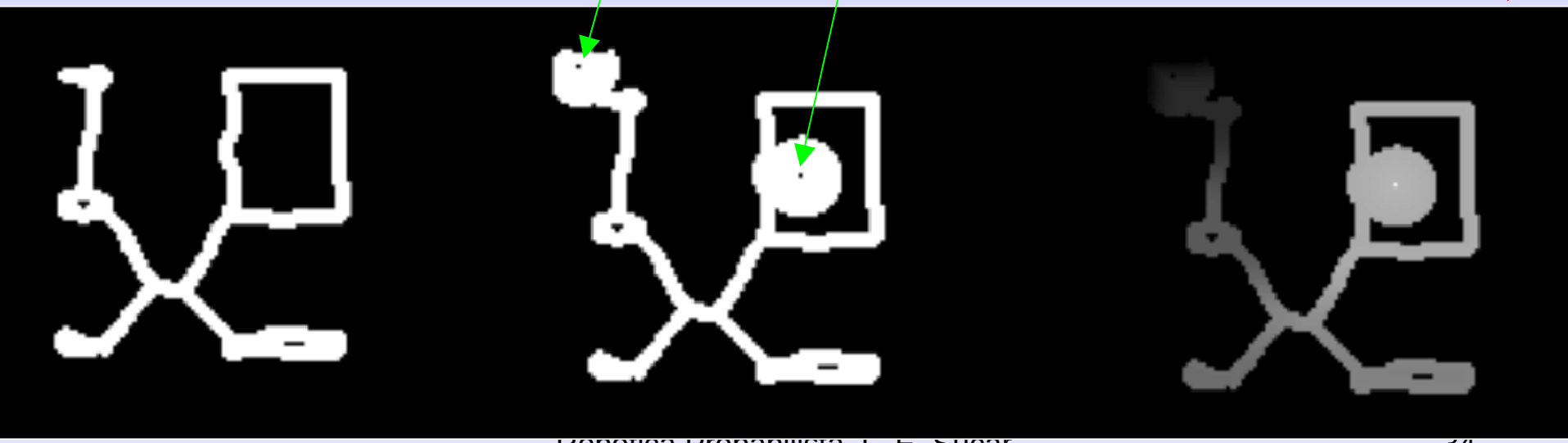


Ejemplo – programación dinámica



Ejemplo de programación dinámica

Se considera un conjunto de celdas (road-map) libres de obstáculos obtenidas previamente



Programación dinámica – otros costos

$V_i(x,y)$: Costo acumulado de viaje a la celda más cercana, si el robot está en la celda (x,y) con orientación θ_i

Inicio:

$V_i(x,y)$ = 0 si la celda (x,y) es la meta

= infinito otras celdas (x,y)

Regla de actualización:

$$V_i(x,y) = \min_{m\_j=(dx,dy) \text{ en Mov}} \{V_j(x+dx,y+dy) + C_{mov}((x,y),(dx,dy)) + C_g(i,j)\}$$

(Sig. celda) (avanzar) (girar)

El costo por avanzar incluye:

- *Distancia entre celdas*
- *Costo de la celda destino de acuerdo a su tipo*

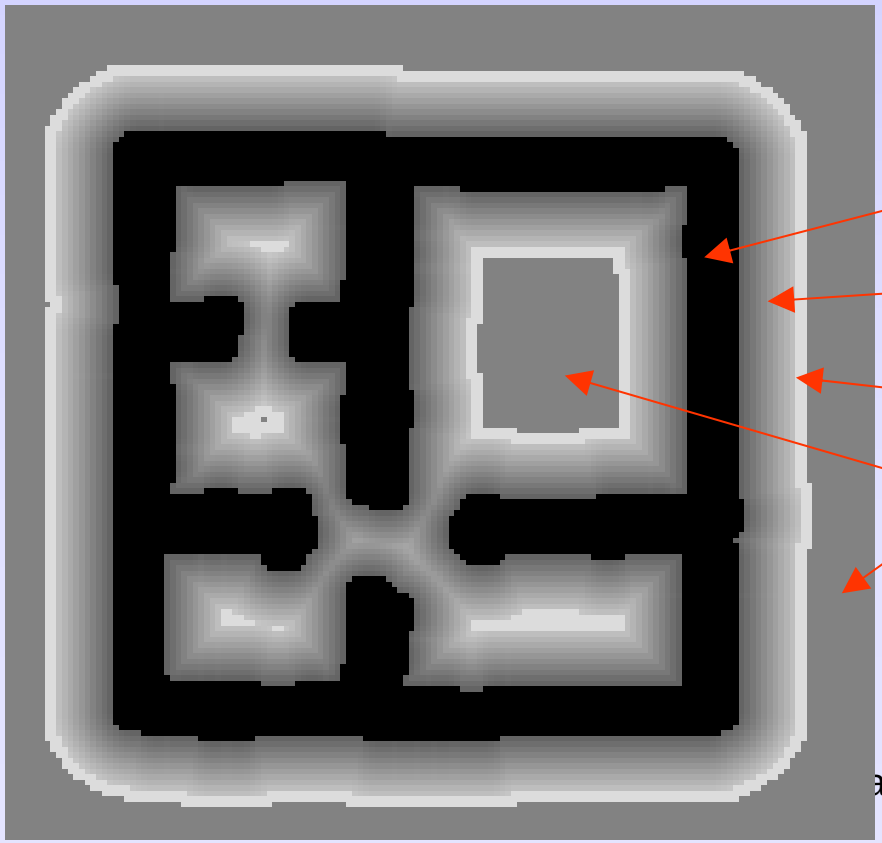
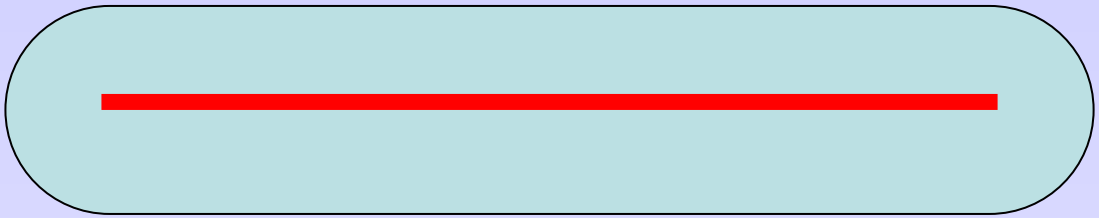
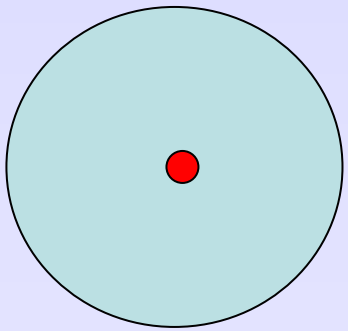


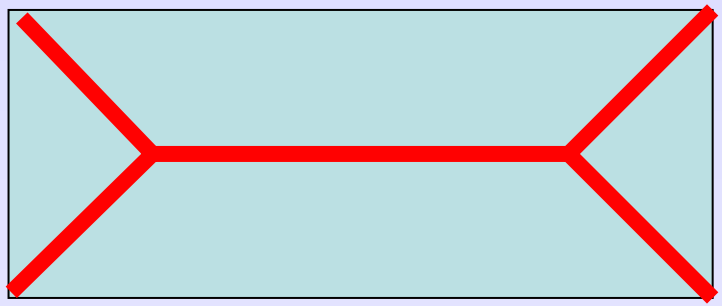
Diagrama de Voronoi

- El diagrama de Voronoi es el conjunto de puntos equidistantes a los límites de 2 o más obstáculos
- Siguiendo el diagrama de Voronoi el robot maximiza la distancia a los obstáculos
- Se puede utilizar un diagrama aproximado considerando una distancia máxima a los obstáculos de forma que se consideren las limitaciones de los sensores

Ejemplos de diagramas de Voronoi

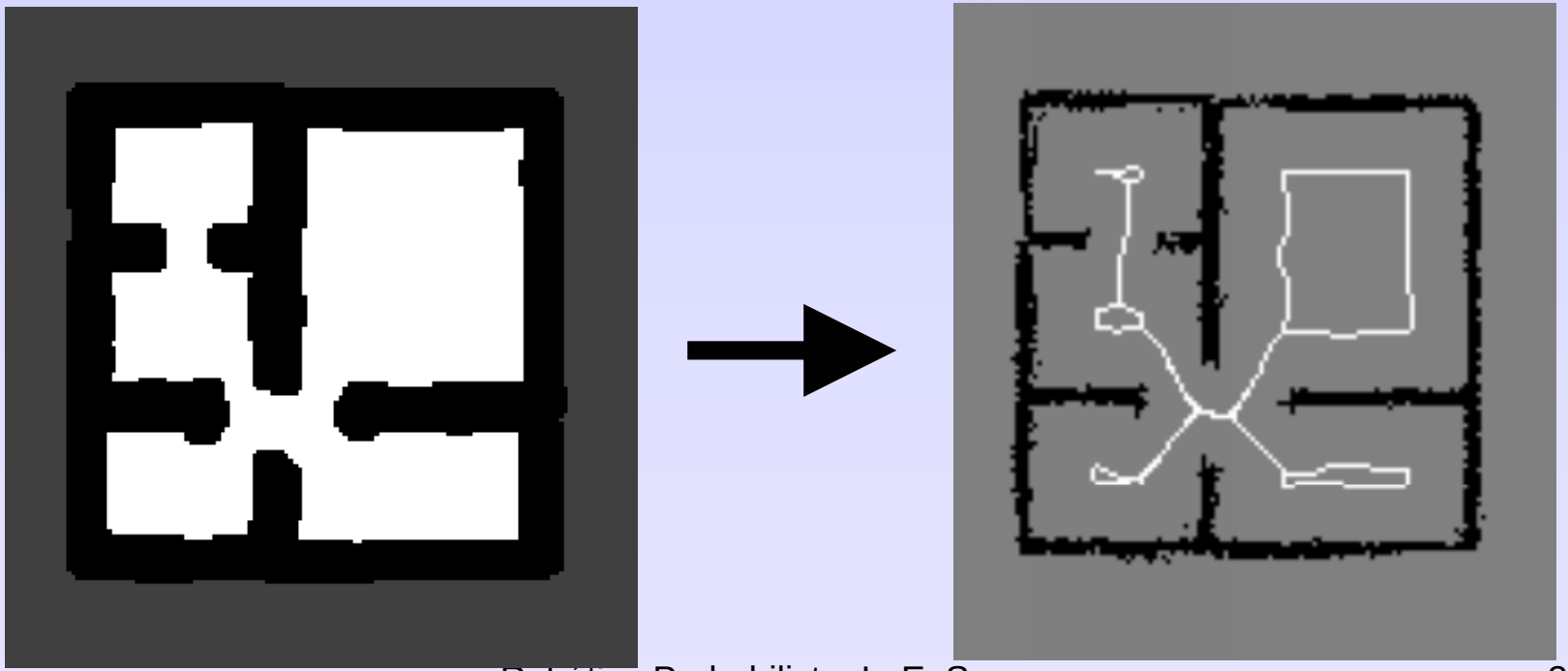






Ejemplo de Diagrama de Voronoi

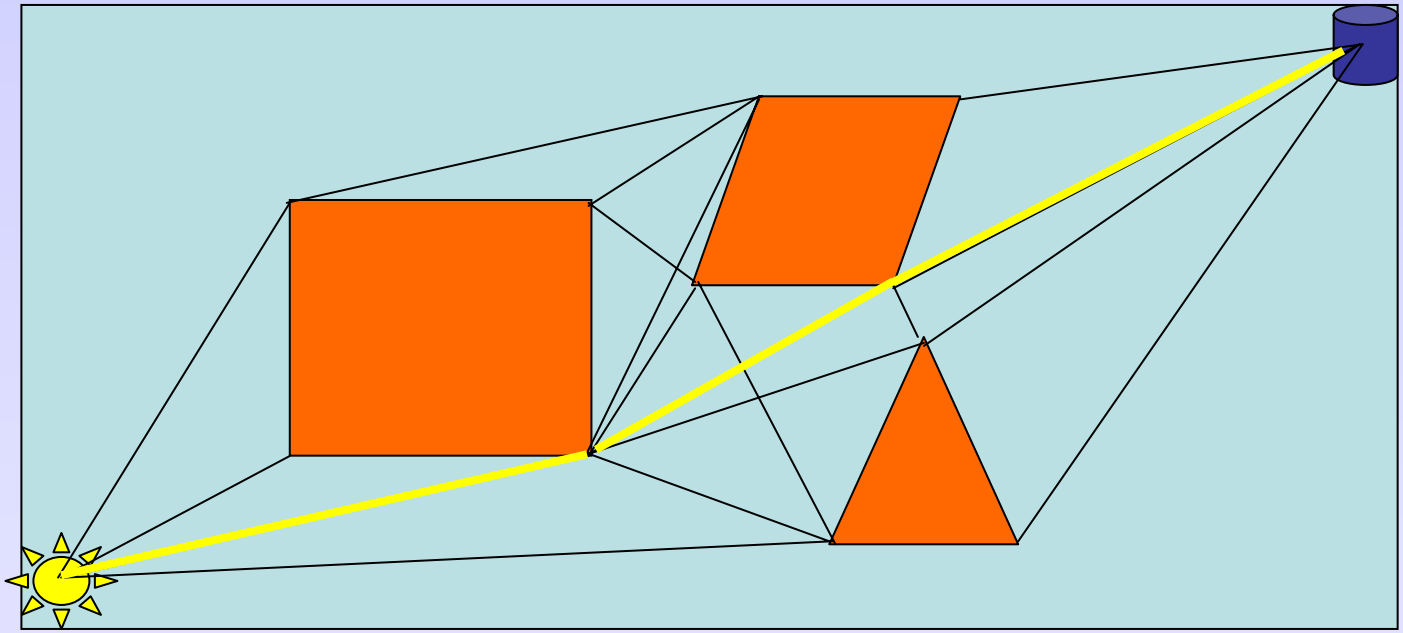
- Izquierda – mapa del espacio libre
- Derecha – diagrama de Voronoi (aproximado)



Gráficas de visibilidad

- Técnica que produce una trayectoria de longitud mínima mediante la solución de un grafo (búsqueda)
- Se basa en la construcción de un grafo de visibilidad, $G = (V, E)$
  - Los vértices, V, corresponden a las esquinas de todos los obstáculos (poligonales), el inicio y la meta
  - Los arcos, E, conectan todos los vértices que son visibles entre sí

Gráficas de visibilidad



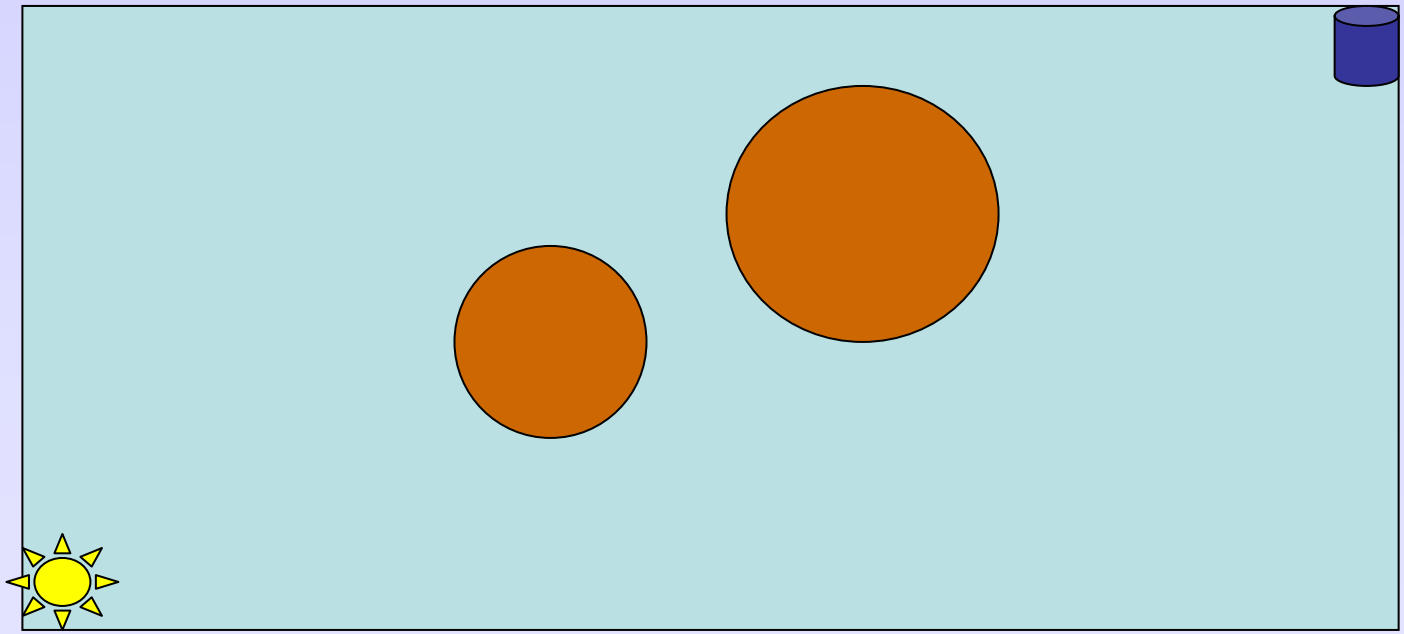
Gráficas de visibilidad

- Existen algoritmos eficientes para construir el grafo de visibilidad y hacer la búsqueda de la trayectoria óptima
- La desventaja es que las trayectorias pasan muy cerca de los obstáculos y consideran un robot puntual – se puede aminorar este problema dilatando a los obstáculos en función del tamaño del robot (problema con robots no cilíndricos)

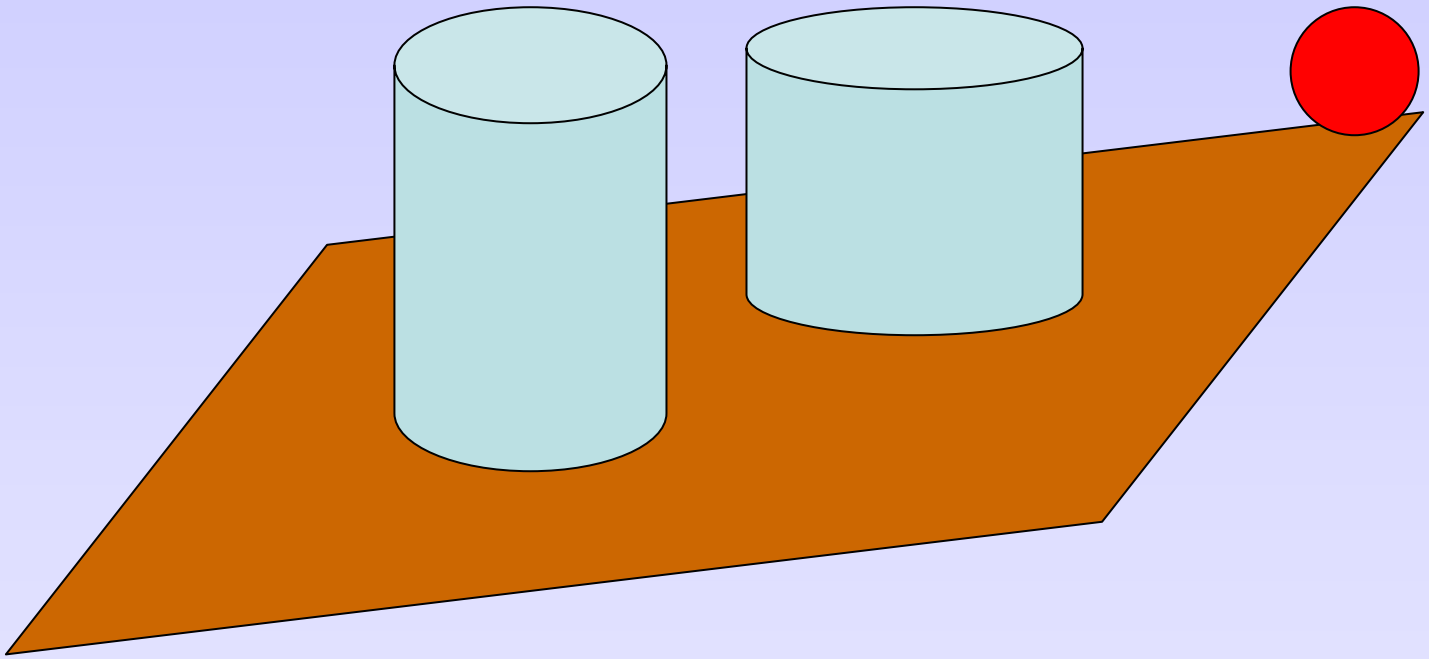
Campos de Potencial

- Se basa en una analogía con campos potenciales eléctricos:
  - El robot se ve como una partícula con carga eléctrica
  - El espacio libre se considera como un campo potencial
  - Los obstáculos tienen un carga eléctrica del mismo signo del robot (se repelen)
  - La meta tiene una carga eléctrica de signo opuesto al robot (se atraen)

Campos de Potencial



Campos de Potencial



Campos de Potencial

- El *campo de potencial diferencial* se construye sumando el campo de la meta, Ug, y el campo de los obstáculos, Uo:

$$U(q) = Ug(q) + \Sigma\, Uo(q)$$

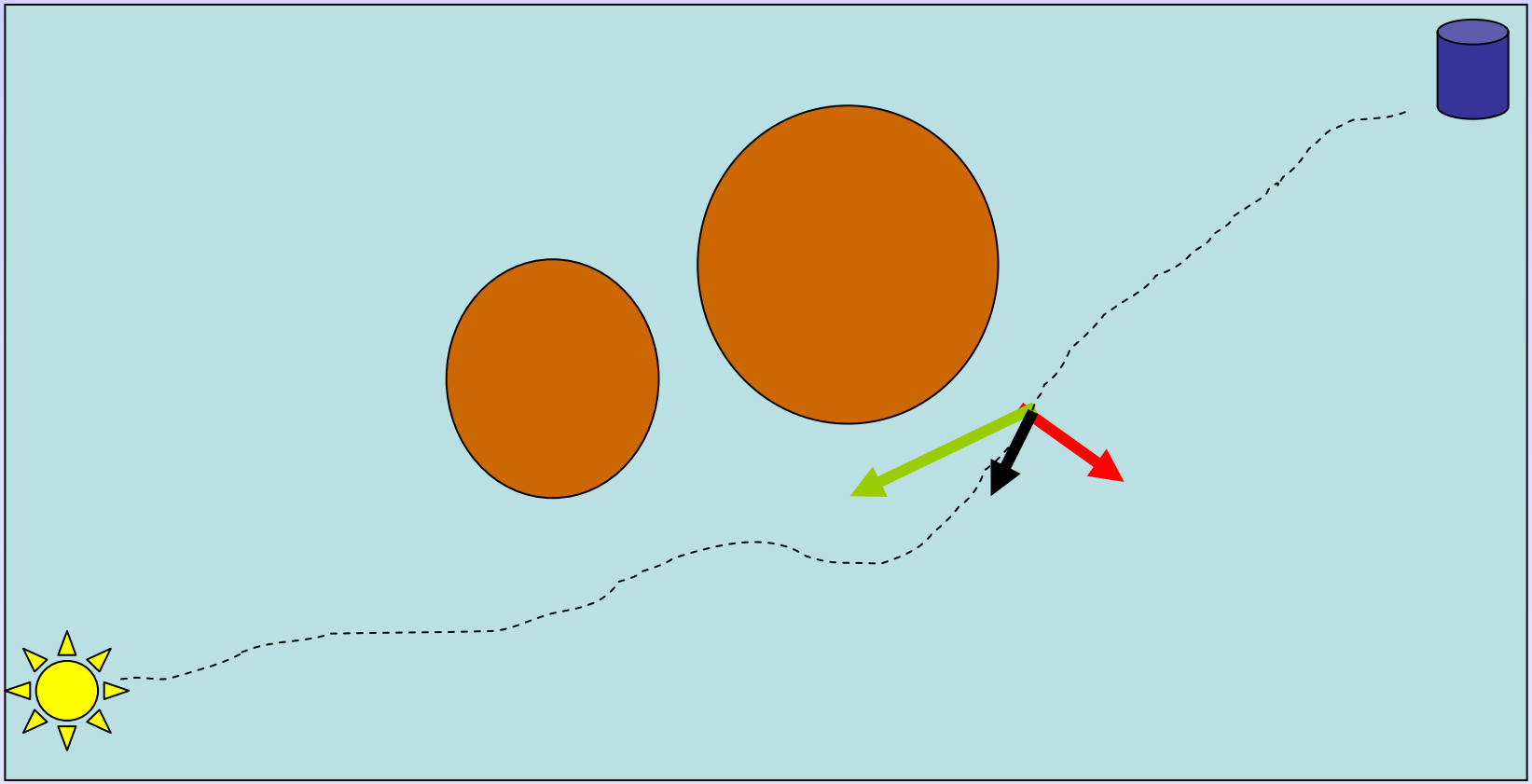
- A partir de este campo se construye un *campo de fuerzas artificial*:

$$F = -\nabla U(q) = \begin{pmatrix} dU/dx \\ dU/dy \end{pmatrix}$$

Campos de Potencial

- Una vez construido el campo de fuerzas, el robot se mueve en función de la *fuerza local* (similar al caso de los valores en programación dinámica)
- Con esto se tiene un esquema robusto que implícitamente tiene un plan de cualquier punto del espacio a la meta

Campos de Fuerzas



Funciones de Potencial

- Para obtener las fuerzas hay que modelar el las funciones de potencial de la meta y obstáculos – calculando el potencial para cada punto del espacio libre:
- Meta – "atractor parabólico"

$$Ug(q) = k_1 \, dist(q, meta)^2$$

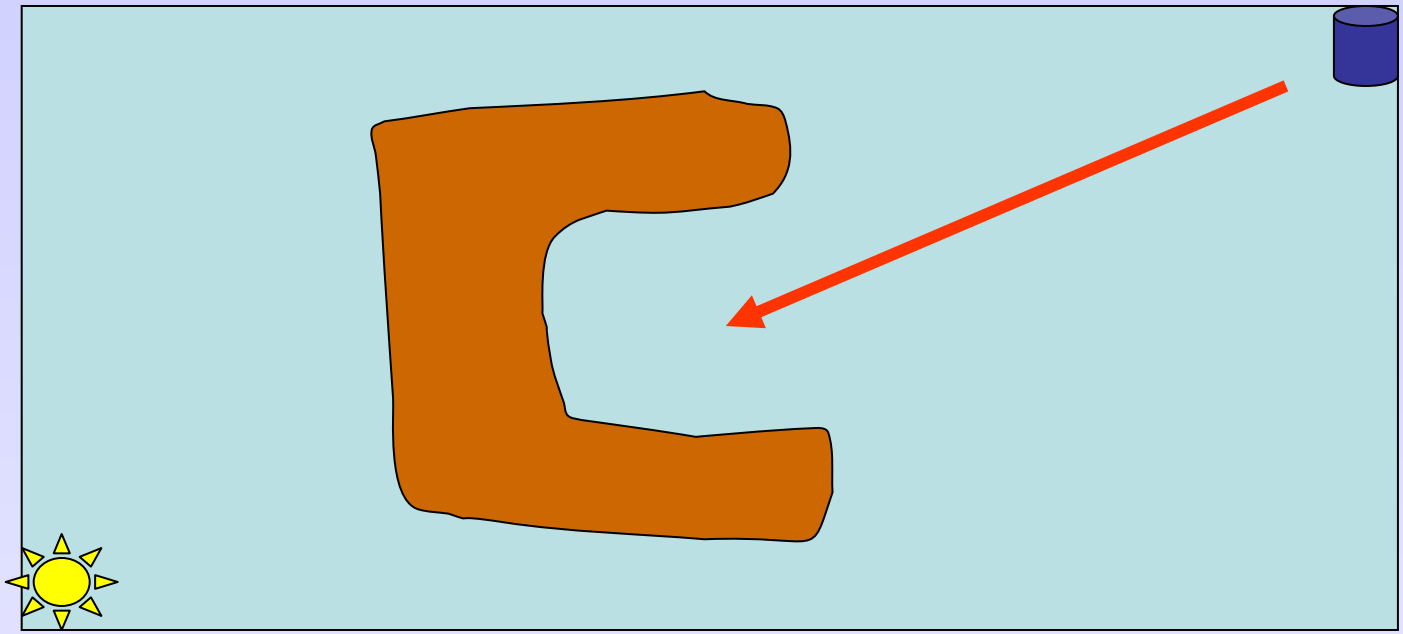
- Obstáculo – "barrera potencial exponencial"

$$Uo(q) = k_2 \, dist(q, obstáculo)^{-1}$$

Funciones de Potencial

- Ventajas:
  - Se pueden generar trayectorias en tiempo real a partir del campo de fuerzas
  - Las trayectorias generadas son suaves
  - Facilita el acoplar la parte de planeación con control
- Desventajas:
  - Mínimos locales!

Trampas (mínimos locales)



Mínimos Locales

- Estrategias para evitar *trampas*:
  - *Backtracking*
  - Movimientos aleatorios
  - Usar otra estrategia, como seguimiento de paredes
  - Aumentar el potencial repulsivo de regiones visitadas

Referencias

- [Dudek, Jenkin] – Cap 5
- [Siegwart et al.] – Cap 6
- [Thrun et al.] – Cap 6
- Romero et al., *Exploration and navigation for mobile robots with perceptual limitations*, IJARS, 2006

Actividades

- Práctica 3: Mapas y Planeación
  - Representa el ambiente mediante un mapa de celdas de ocupación (dilatar obstáculos)
  - En base a dicho mapa desarrolla un algoritmo para encontrar una ruta de una celda inicial a una final en base a programación dinámica
  - Haz que el robot “ejecute” la ruta encontrada
  - Prueba tu sistema en diferentes ambientes con diferentes posiciones iniciales y finales
  - Opcional: incorpora “costos” a las celdas de acuerdo a su distancia a los obstáculos (u otros factores) y repite el mismo procedimiento