Instituto Politécnico Nacional

Escuela Superior de Cómputo

Estructuras de Datos

Tema 07: Backtraking

M. en C. Edgardo Adrián Franco Martínez
http://www.eafranco.com
edfrancom@ipn.mx
@edfrancom edgardoadrianfrancom

Contenido

- Backtraking
  - Búsqueda en profundidad
  - Backtracking O(Complejidad)
  - Ejemplo: Problema de las N reinas
- Backtracking en paralelo
- Backtraking ejemplo del relleno con color
  - Solución iterativa backtraking con una pila
  - Solución backtraking recursiva
- Otros problemas donde se aplica backtraking
- Backtraking (Resumen)
- Backtracking paralelo
  - Asignación estática
  - Asignación dinámica

Backtracking

- **Backtracking *(vuelta atrás)*** es una estrategia para encontrar soluciones a problemas que satisfacen restricciones. El término "backtrack" fue acuñado por primera vez por el matemático estadounidense D. H. Lehmer en la década de 1950.

  - **El principio de backtracking (vuelta atrás)** se suele aplicar en la resolución de un gran número de problemas, muy especialmente en juegos y problemas de optimización.

  - El backtraking realiza una **búsqueda exhaustiva y sistemática en el espacio de soluciones del problema**.

  - Se utilizan para resolver problemas para los que no existe un algoritmo eficiente para resolverlos.

  - Mediante la programación paralela se intenta reducir el tiempo de ejecución de estos algoritmos

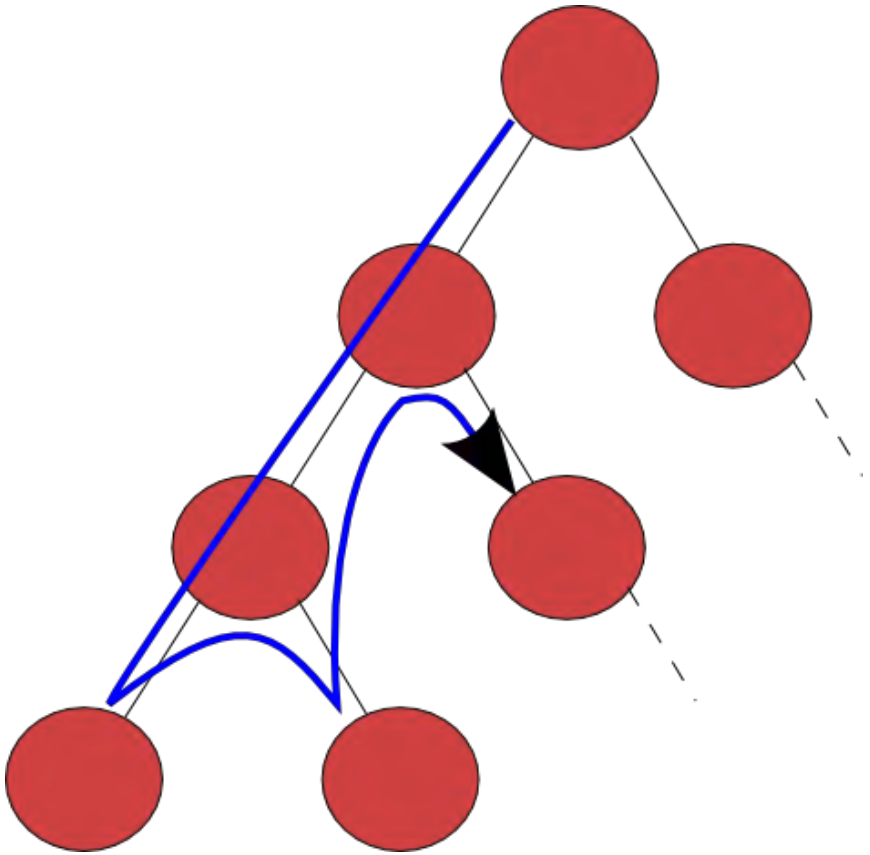
- **Backtracking** se asemeja a un recorrido en profundidad dentro de un grafo dirigido y acíclico.

- Una solución mediante **backtraking** construye soluciones parciales a medida que progresa el recorrido; estas soluciones parciales limitan las regiones en las que se puede encontrar una solución completa.

- El recorrido tiene éxito si, procediendo de esta forma, se puede definir por completo una solución. En este caso el algoritmo puede detenerse (si lo único que se necesita es una solución del problema) o seguir buscando soluciones alternativas (si deseamos examinarlas todas).

- Si el recorrido no tiene éxito en alguna etapa y la solución parcial construida hasta el momento no se puede completar. El recorrido vuelve atrás exactamente igual que en un recorrido en profundidad, eliminando sobre la marcha los elementos que se hubieran añadido en cada fase. Cuando vuelve a un nodo que tiene uno o más vecinos sin explorar y prosigue el recorrido de una solución.

- No siguen unas reglas para la búsqueda de la solución simplemente una búsqueda sistemática, que más o menos viene a significar que hay que probar todo lo posible hasta encontrar la solución o encontrar que no existe solución al problema.



- El **backtracking** es una **técnica de programación** para hacer **búsqueda sistemática** a través de **todas las configuraciones** posibles dentro de un **espacio de búsqueda**.

- Para lograr esto, los algoritmos de tipo backtracking construyen posibles soluciones candidatas de manera sistemática. En general, dado una solución candidata **s**.

1. **Verifican si s es solución:** Si lo es, hacen algo con ella (depende del problema).

2. **Construyen todas las posibles extensiones de s**, e invocan recursivamente al algoritmo con todas ellas.

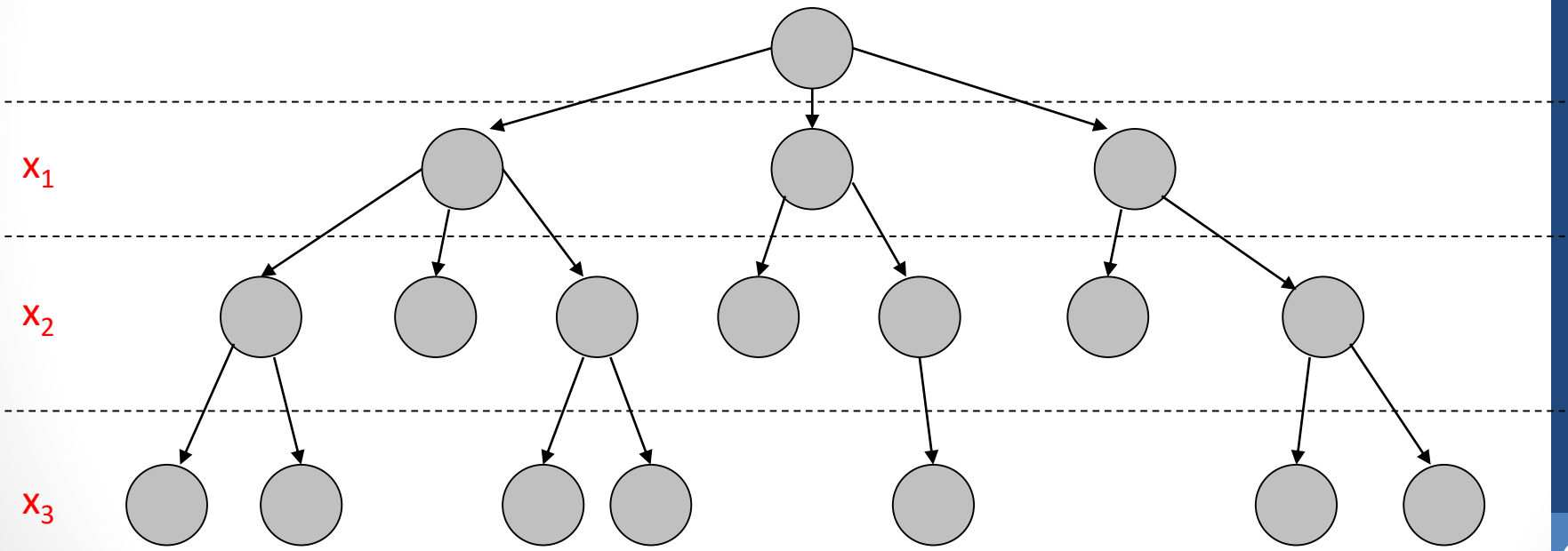
- A veces los algoritmos de tipo backtracking se usan para encontrar una solución, pero otras veces interesa que las revisen todas (por ejemplo, para encontrar la más corta).

- La solución de un problema de backtracking se puede expresar como una **tupla (x_1,x_2,…,x_n),** que satisface una restricciones **R(x_1,x_2,…,x_n)** y a veces optimizando una **función objetivo**.

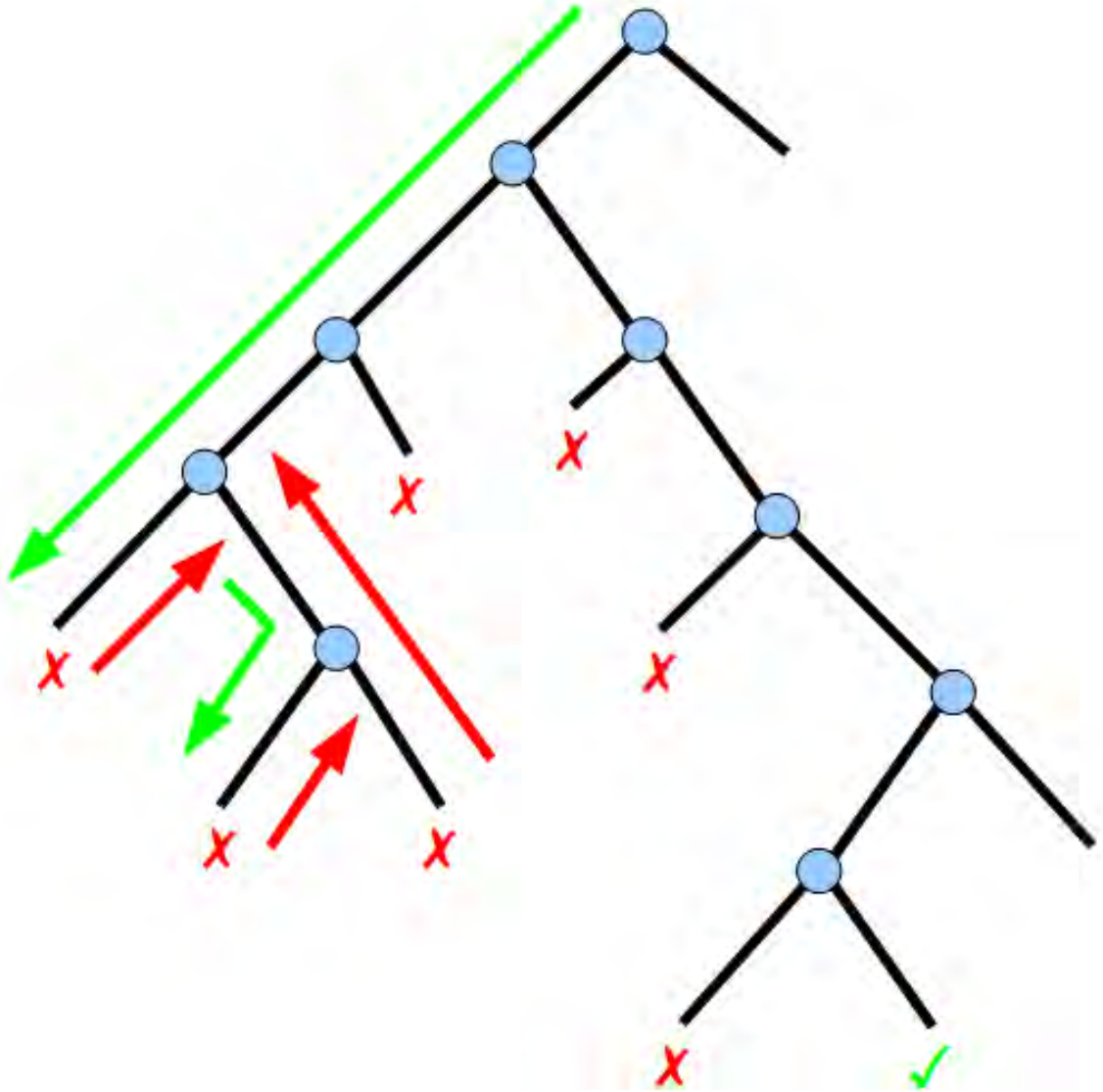
- En cada momento el algoritmo se encontrará en un cierto nivel k, con una solución parcial (x_1,x_2,…,x_k) (con k<=n).
  - Si puede añadirse un elemento x_{k+1} a la solución parcial se avanza al nivel k+1.
  - Si no se prueban otros valores válidos para x_k.
  - Si no existe ningún valor que sea válido por probar, se retrocede al nivel anterior k-1.
  - Se continua con este proceso hasta que la solución parcial sea una solución del problema o hasta que no queden más posibilidades por probar (en el caso de que no se encuentre ninguna solución o se busquen todas las soluciones del problema).

Búsqueda en profundidad

- Se realiza una búsqueda en profundidad en el árbol de soluciones del problema.



- Se realiza una búsqueda en profundidad en el árbol de soluciones del problema.



Backtracking **O**(Complejidad)

- Para realizar una **búsqueda exhaustiva** en el **espacio de soluciones del problema**, los algoritmos de backtracking son bastante **ineficientes**.

- En general, se tienen tiempos con órdenes de complejidad factoriales o exponenciales ***O(Exp n)*.**

- Por esto, los algoritmos de backtracking se utilizan en problemas para los que no existen un algoritmo eficiente que los resuelva.

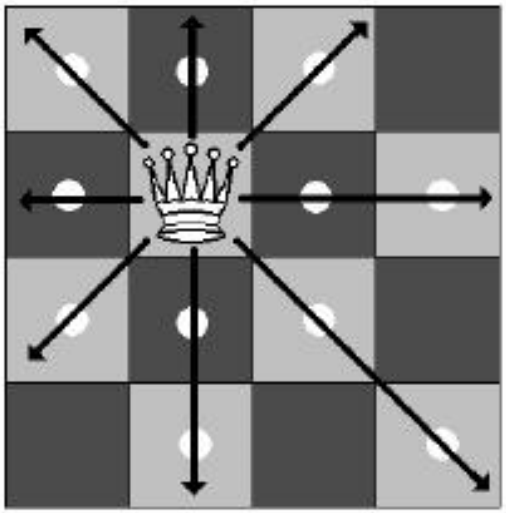
Ejemplo (Problema N-Reinas)

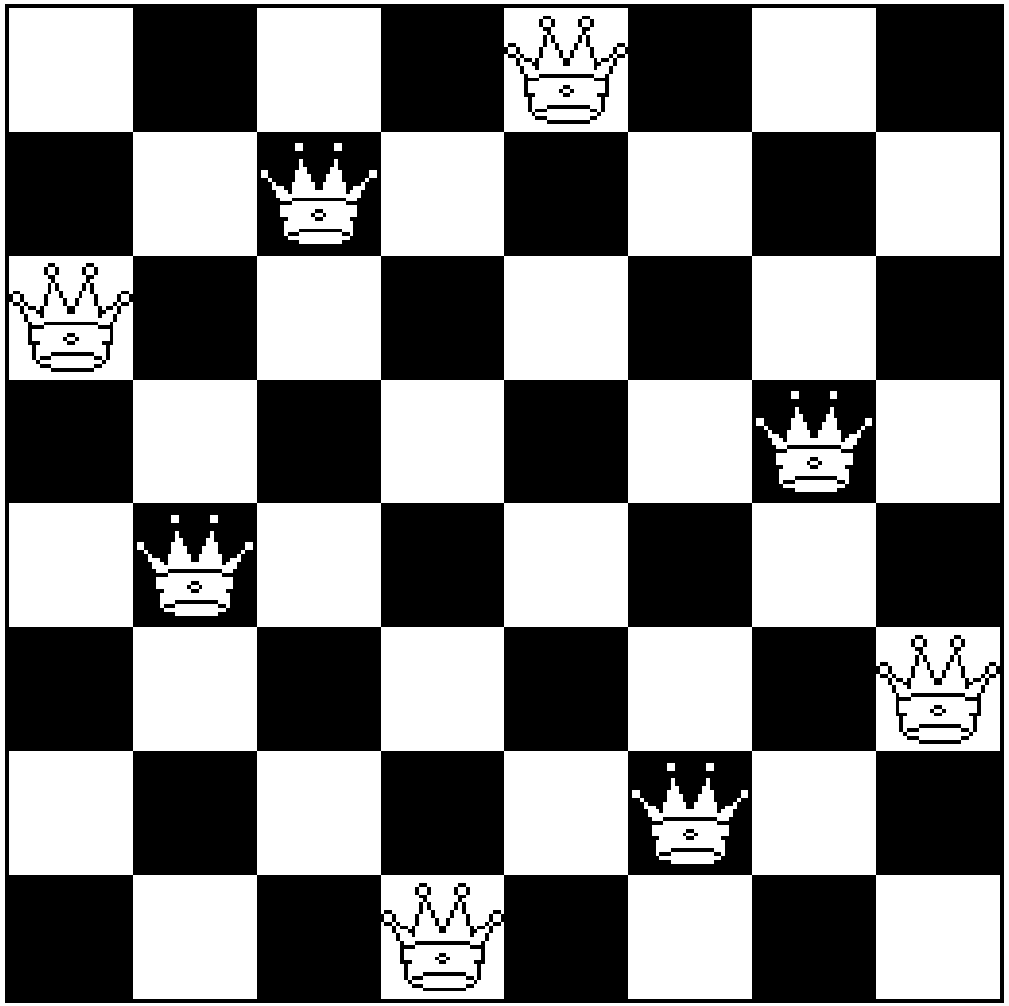
- El problema de las N-Reinas consiste en colocar **n** reinas en un tablero de ajedrez de tamaño **n*n** de forma la reinas no se amenacen según las normas del ajedrez. Se busca encontrar una solución o todas las soluciones posibles.

- Este problema puede resolverse utilizando un esquema de backtracking.

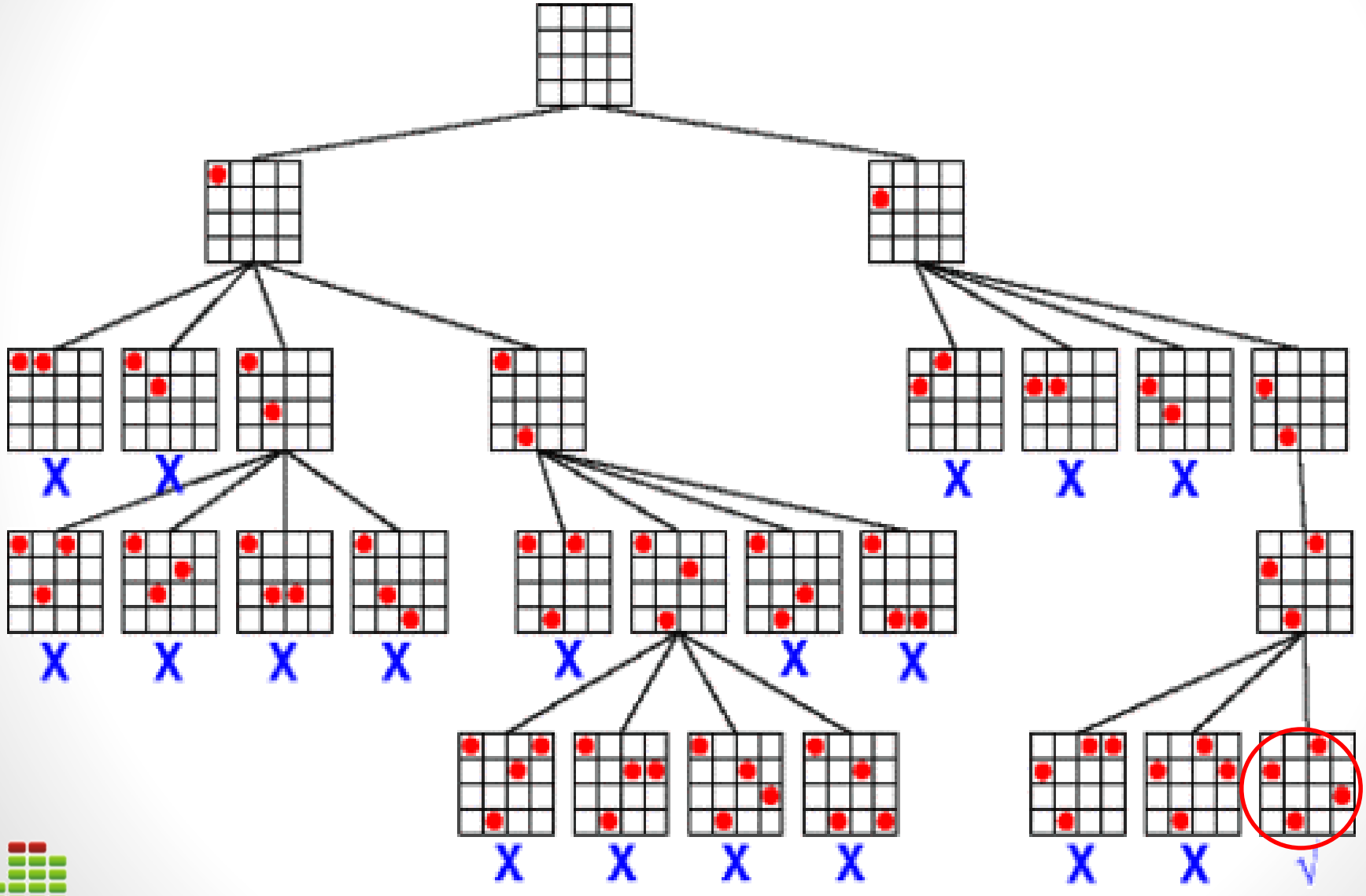
- Cualquier solución del problema estará formada por una n-tupla $(x_1,x_2,\ldots,x_n)$, dónde cada x_i indica la columna donde la reina de la fila i-ésima es colocada.

- Las restricciones para este problema consisten en que dos reinas no pueden colocarse en la misma fila, ni en la misma columna ni en la misma diagonal.

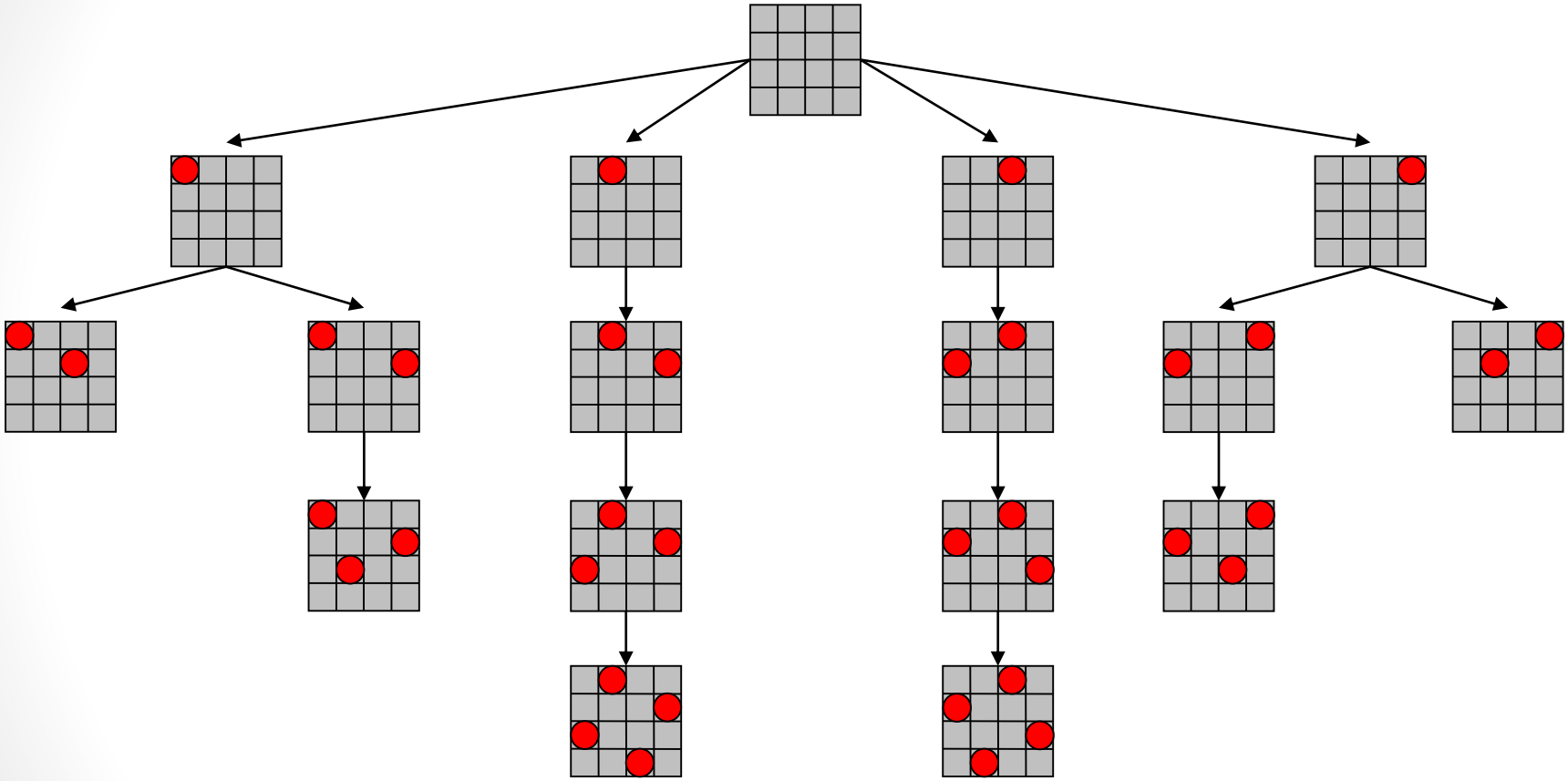




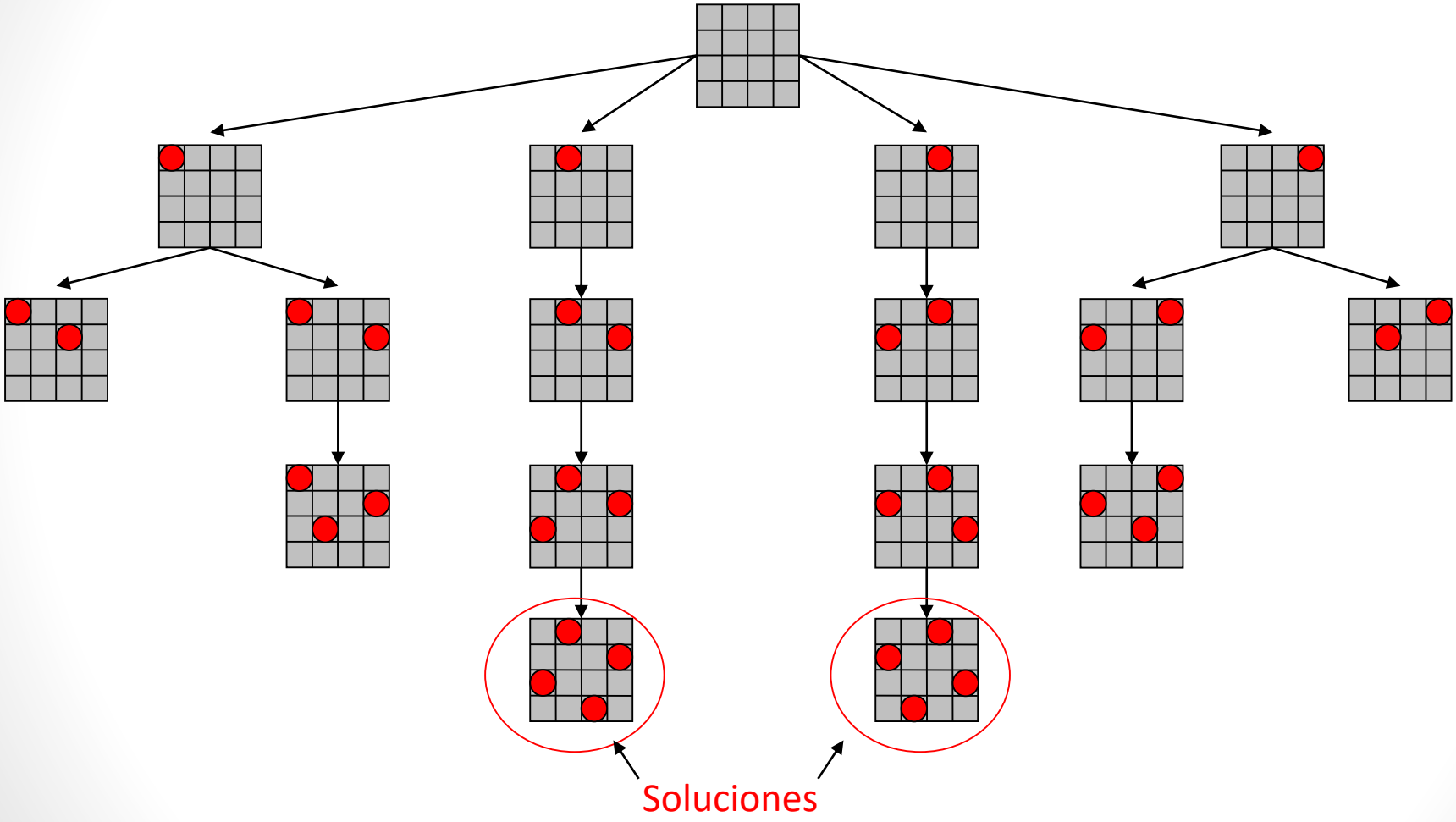
- Análisis para encontrar **una** solución al problema de las 4 reinas.



- Por ejemplo, el problema de las 4-Reinas tiene dos posibles soluciones: [2,4,1,3] y [3,1,4,2].

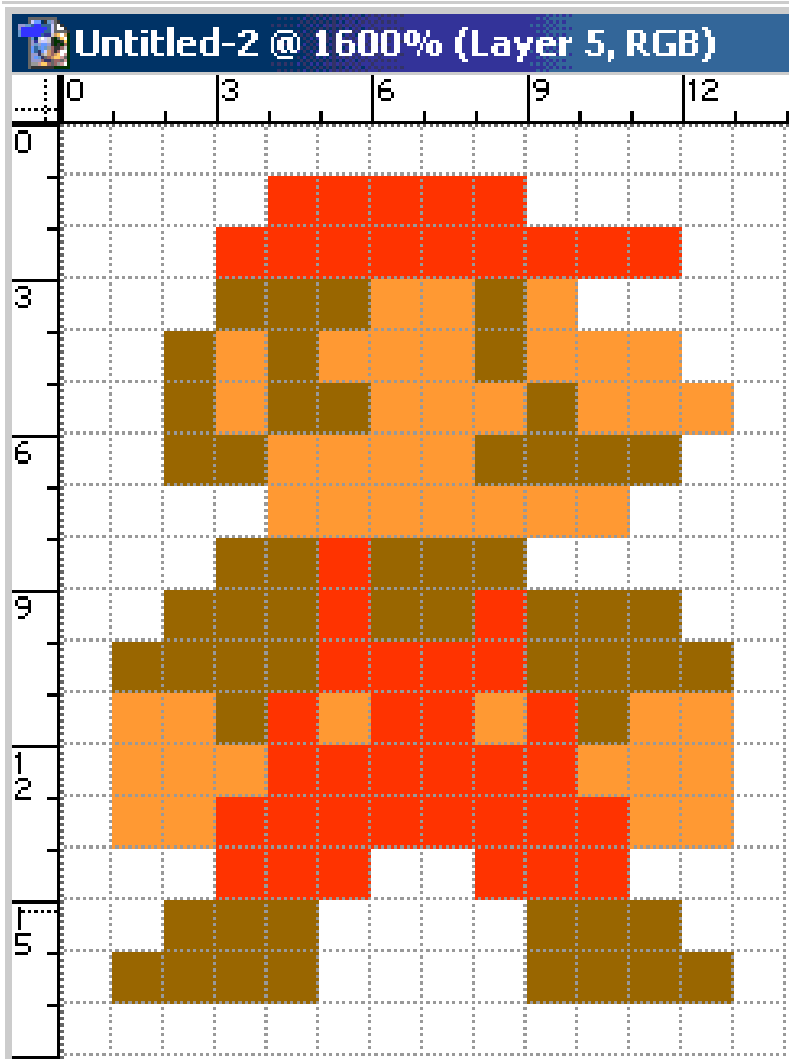


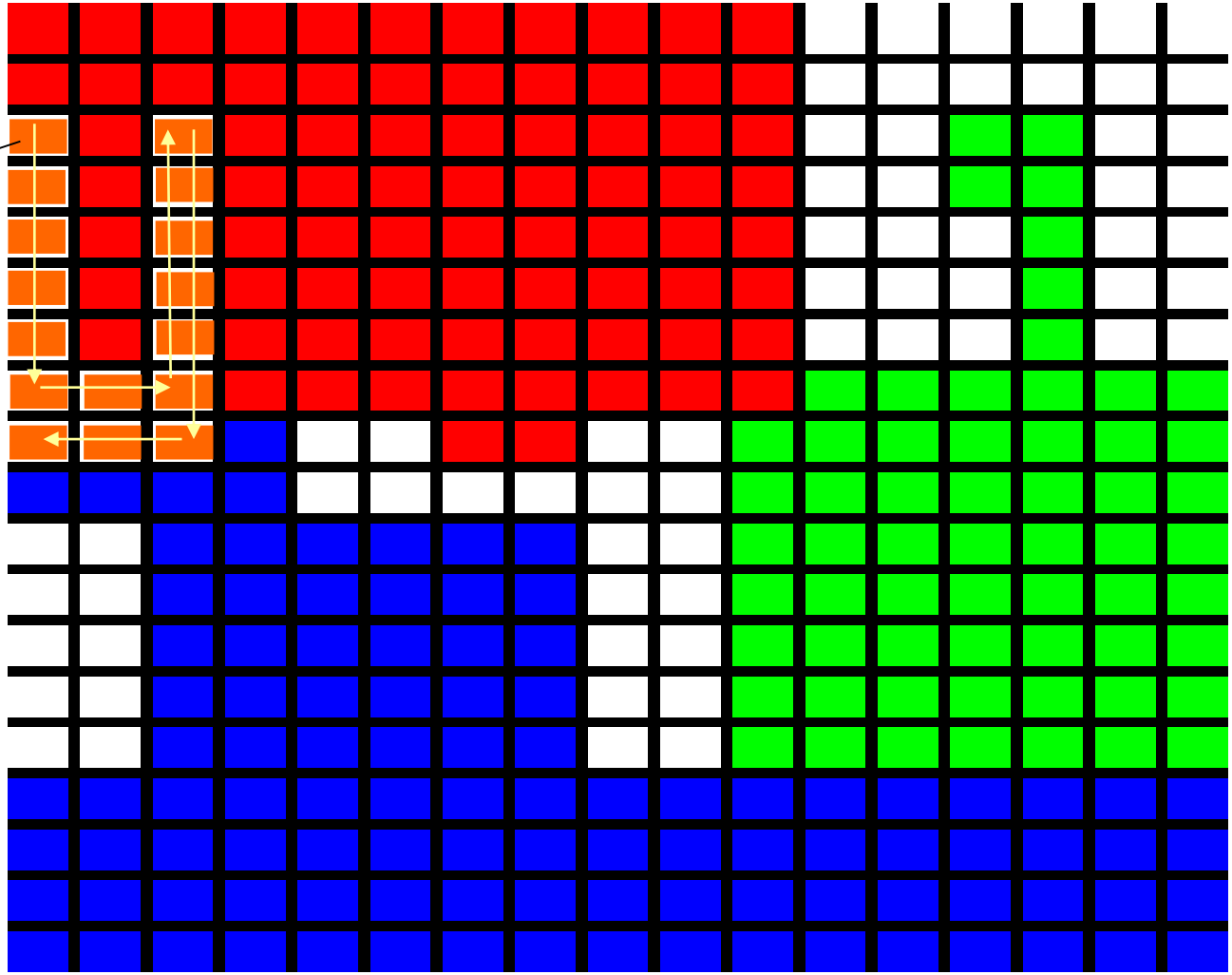
- Por ejemplo, el problema de las 4-Reinas tiene dos posibles soluciones: [2,4,1,3] y [3,1,4,2].

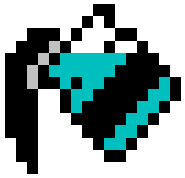


Backtraking ejemplo del relleno con color

- Una imagen es una matriz de valores de intensidad de color y/o niveles de gris.




Pintar
aquí



- El programa necesita que le indiquen de que color desean rellenar y desde donde
  - Color y Posición (fila, columna): Naranja y (2,0)
  - Obtiene también el color de la Posición escogida antes de pintarla (Blanco en nuestro ejemplo)

- La idea es cambiar todos los cuadros Blancos adyacentes por Naranjas.

Solución iterativa relleno con color con una pila

Relleno(Posicion_inicial)

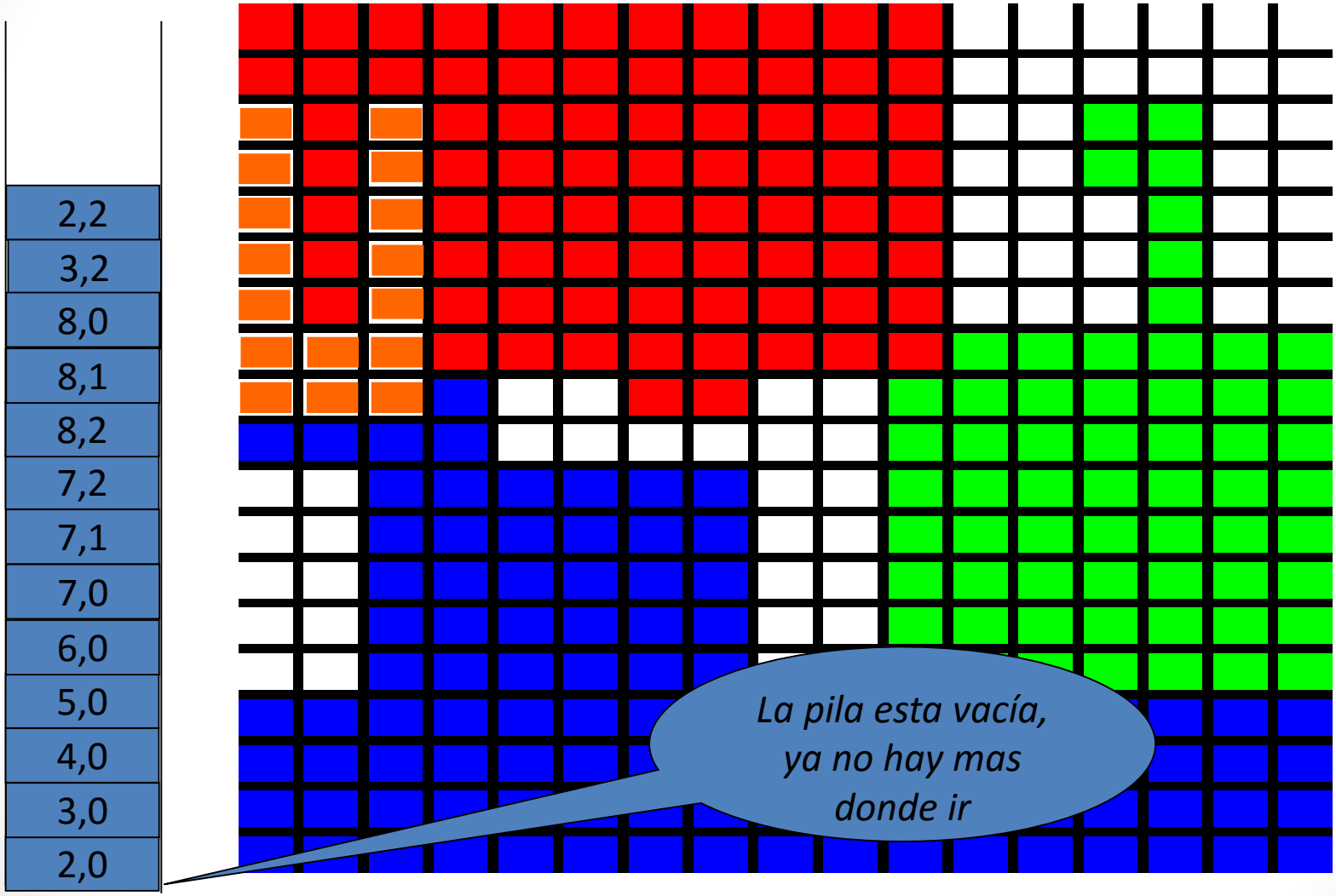
- Si la Posición inicial dada es Blanca
  - La pinto de Naranja
  - Guardo en una pila el rastro de lo ultimo pintado
    - En caso de que necesite **regresar** a este punto mas tarde

- Ahora en cada una de las 4 posibilidades
  - Arriba, Derecha, Abajo, Izquierda

- Pregunto si puedo moverme (si hay posición y si es de color blanco)
  - Si puedo ir a otro cajón me muevo y se ***repite todo***
  - Si no puedo ir, obtengo la ultima posición pintada **(de mi pila de rastros)**
    - Y repito todo (vuelvo a intentar)
    - Si ya no hay rastros guardados, es que no hay **mas nada que pintar**
      - **Todo termina**

Problema del relleno con color

Solución iterativa backtraking con una pila



Solución recursiva relleno con color

Relleno (posición)

- Si la Posición dada es Blanca
  - La pinto de Naranja
- Ahora en cada una de las 4 posibilidades
  - Arriba, Derecha, Abajo, Izquierda
- Pregunto si puedo moverme (si hay posición y si es de color blanco)
  - Si puedo ir a otro cajón me muevo y vuelvo a llamar a **Relleno(Posición posible)**
- Si no hay mas posiciones posibles termino

Otros ejemplos de fácil solución con backtraking

- Dado un laberinto, determine la ruta para llegar del inicio al fin.
- Dada la matriz de adyacencia de un mapa de ciudades, determine si hay o no camino entre dos ciudades dadas.
- Otros
  - Formación de una palabra con n cubos
  - Sudoku
  - Problema del dominó
  - Problema de los movimientos de un caballo
  - Problema del reparto del botín
  - Problema de la mochila utilizando
  - Problema del viajante sobre grafos dirigidos

Backtraking (Resumen)

- **Método para resolución de problemas**
  - Backtracking → Retroceso o Vuelta Atrás
- **Realiza una búsqueda exhaustiva de una posible solución**
  - Consiste en seguir un camino buscando una solución
  - Si por ese camino no se llega a la solución, se retrocede por el camino seguido hasta que se encuentre otro camino
  - O hasta que se llegue al inicio, lo cual indica que ya no hay solución
- **Como se implementa**
  - **Con Recursividad**
  - **Con Pilas Dinámicas**

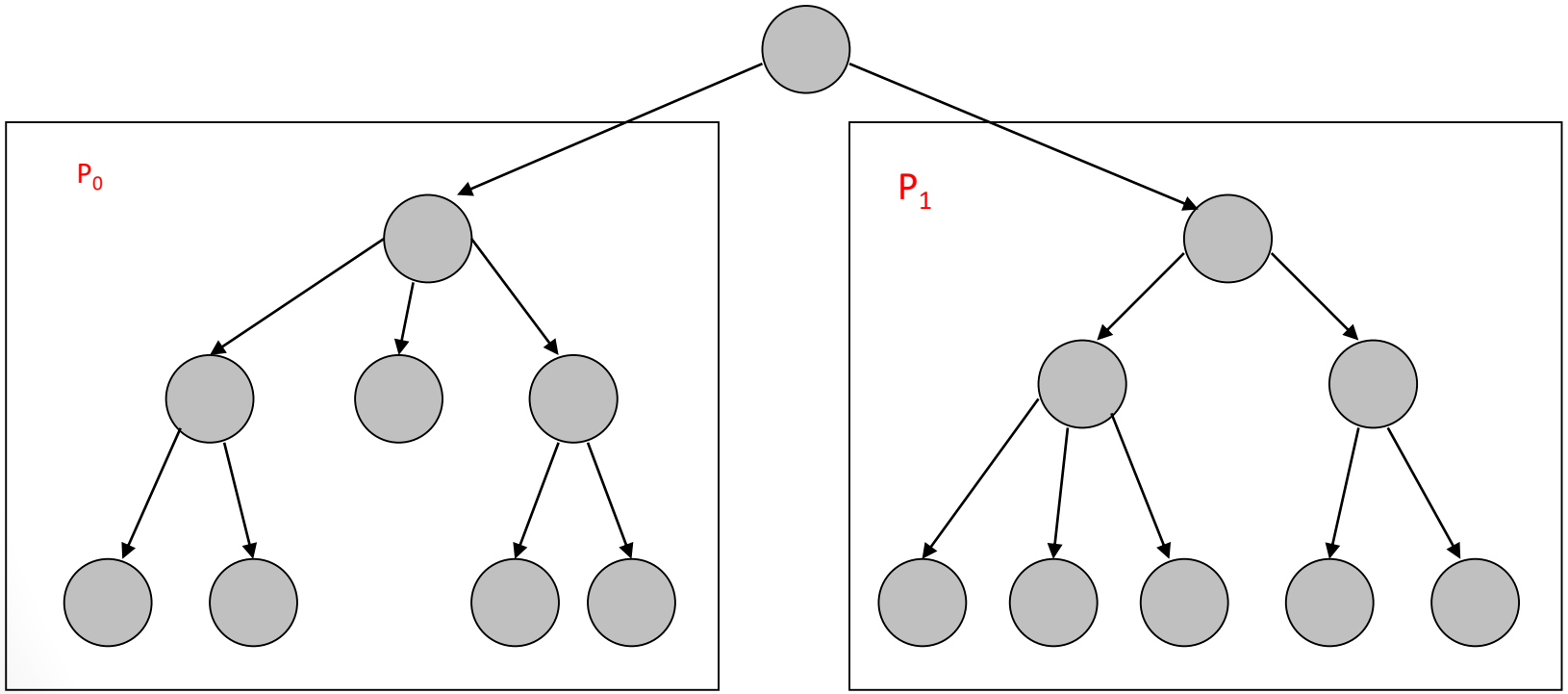
Backtracking paralelo

- Se intenta distribuir el espacio de búsqueda entre los distintos procesadores, de forma que cada uno busque la solución del problema en un subespacio de soluciones distinto.

- Así se exploran varias ramas del árbol de soluciones al mismo tiempo por distintos procesadores.

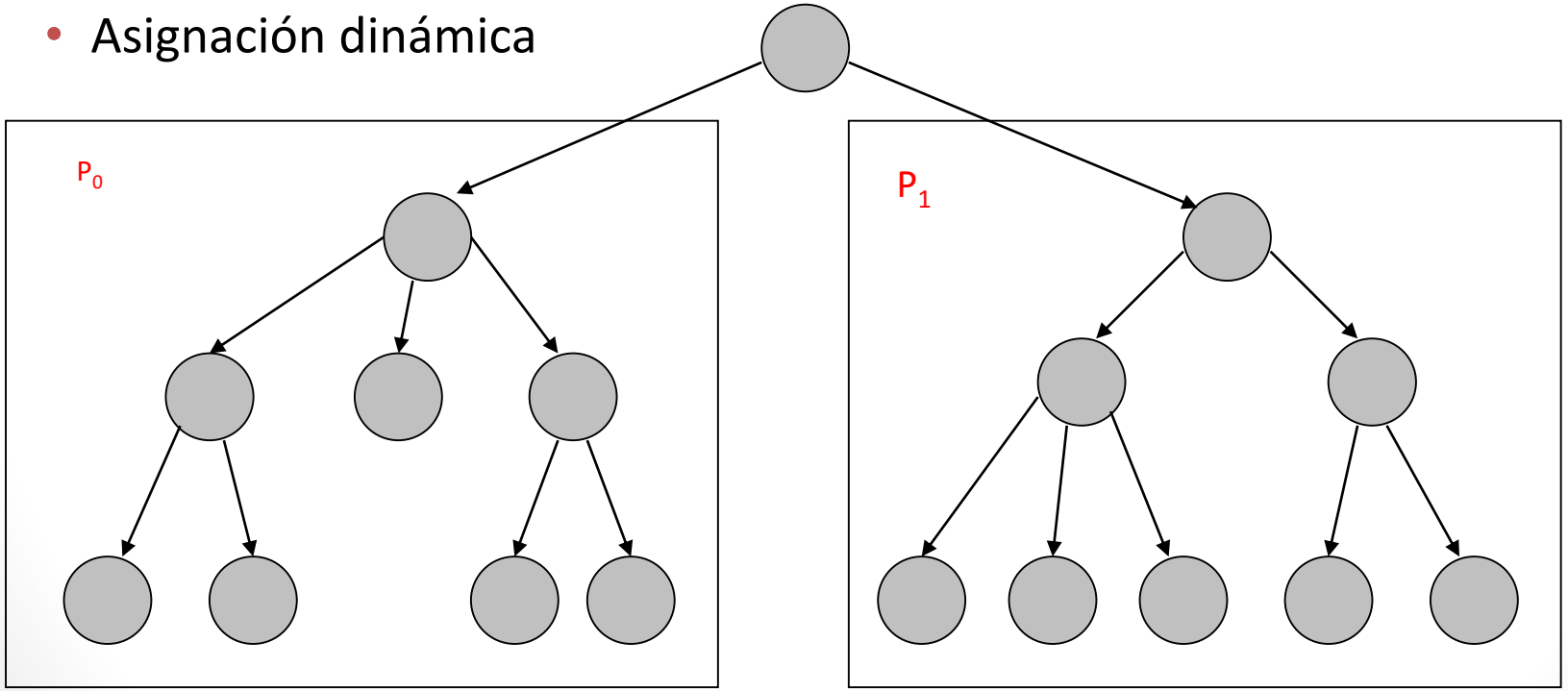
- Aumentan las posibilidades de encontrar la solución del problema en menor tiempo.

Backtracking paralelo



- Un factor crítico a la hora de formular un algoritmo de backtracking en paralelo es cómo se realizará la distribución del espacio de búsqueda entre los distintos procesadores.

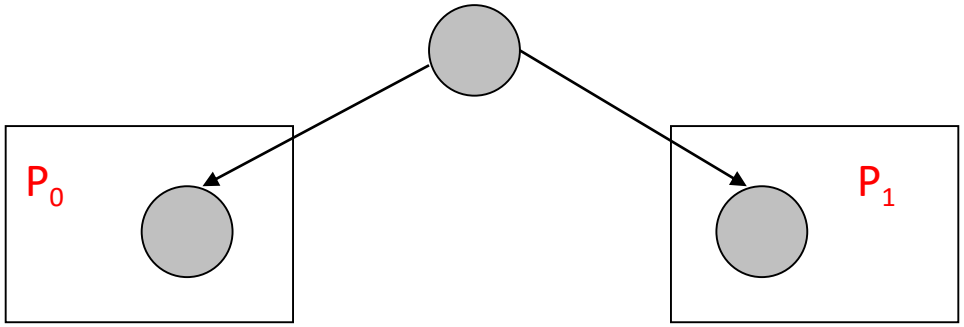
- Dos alternativas:
  - Asignación estática
  - Asignación dinámica

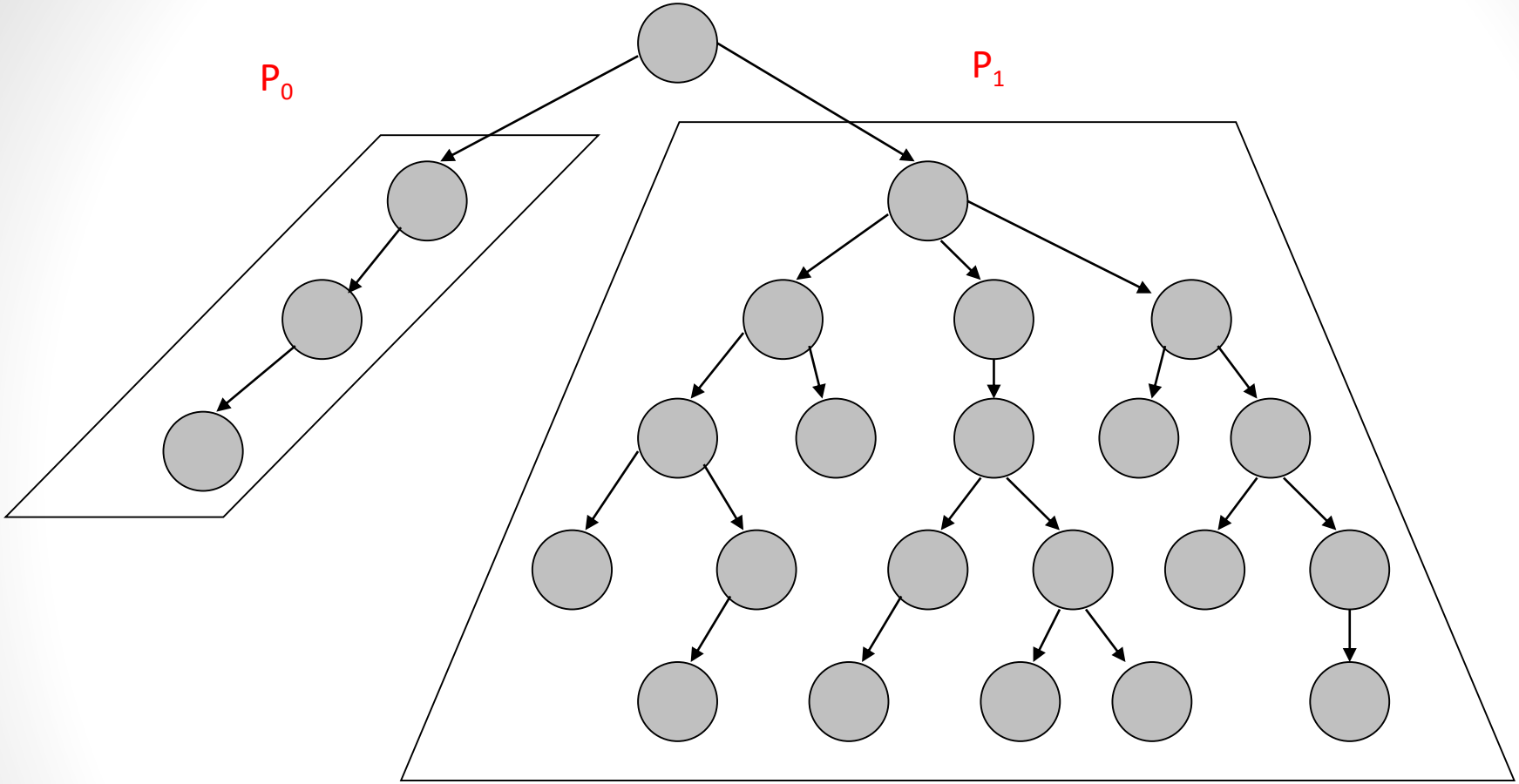


Backtracking Paralelo (Asignación estática)

- **Asignación estática**
  - Un procesador expande el nodo raíz y genera varios nodos.
  - A cada procesador se le asigna un número de estos nodos.
  - Así se divide el espacio de búsqueda global entre los distintos procesadores.
  - Cada procesador comienza a buscar la solución en el espacio de búsqueda que le corresponda, siguiendo el proceso normal de backtracking.
  - No se necesitan comunicaciones entre procesos.

- **Problemas**
  - No se sabe a priori si se divide el espacio de búsqueda equitativamente entre los distintos procesadores.
  - Puede haber desequilibrio entre la carga de trabajo de los procesadores.
  - A un procesador puede que se le asigne poco trabajo y este la mayor parte de tiempo inactivo.


P_0
P_1


P_0
P_1

Backtracking Paralelo (Asignación dinámica)

- Cada procesador trabaja en una parte del espacio de búsqueda.
- Cuando un procesador termina de trabajar, solicita más trabajo a otro procesador que tiene más trabajo.
- Proceso:
  - Cada procesador dispone una pila en su espacio local para guardar nodos no expandidos.
  - Cuando un procesador expande un nodo, los nuevos nodos que se crean se introducen en la pila.
  - Cuando la pila de un procesador esta vacía, el procesador solicita nodos de la pila de otro procesador.

- Al principio, el espacio total de búsqueda es asignado a un procesador y a los demás procesadores no se les asigna ningún trabajo.

- Ese procesador será el encargado de distribuir inicialmente el trabajo cuando los demás procesadores se lo soliciten.

- A los procesos que envían trabajo se les denomina donadores y a los que lo solicitan y lo reciben se les denomina receptores.

- Un procesador puede estar en 2 estados:
  - Activo: Esta realizando algún trabajo.
  - Inactivo: Al procesador no le queda ningún nodo por explorar en su pila y esta solicitando trabajo a otros procesadores.

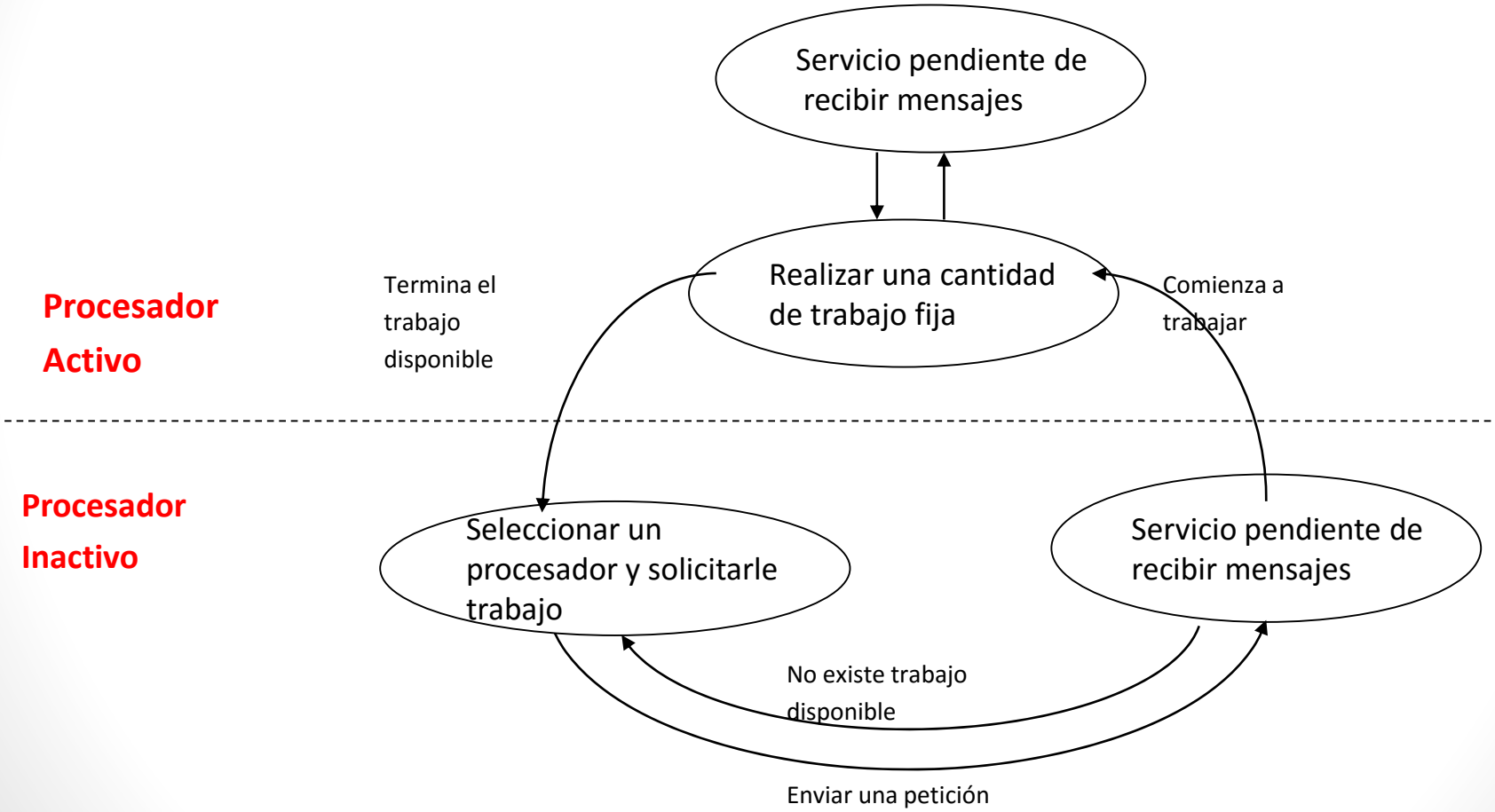
- Estado inactivo:
  - El procesador selecciona un procesador donador y le envía una petición de trabajo.
  - Si recibe trabajo del donador, se activa y comienza a trabajar.
  - Si recibe un mensaje de reject (el procesador donador esta inactivo), selecciona a otro procesador donador para enviarle una petición de trabajo.

- **Estado inactivo:**
  - Este proceso se repite hasta que el procesador recibe trabajo del donador o hasta que todos los procesadores entran en estado inactivo.

- **Estado activo:**
  - El procesador realiza una cantidad de trabajo fijo (expande un nº determinado de nodos).
  - Cuando termina de realizar el trabajo, chequea si existen peticiones de trabajo a cargo de los demás procesadores.
  - Si recibe alguna petición de trabajo, el procesador donador particiona en dos partes su pila de nodos, y envía una parte al procesador que realizo la petición.
  - Cuando el procesador ha recorrido todo su espacio de búsqueda (su pila esta vacía), entra en estado inactivo.

Backtracking Paralelo

Asignación dinámica



- Cuando un procesador encuentra la solución, envía un mensaje de broadcast a todos procesadores para estos paren la búsqueda.

- Si no se recibe este mensaje, el algoritmo terminará cuando todos los procesos estén en estado inactivo.

- Los algoritmos paralelos de búsqueda en un árbol de soluciones tienen 2 características fundamentales que determinan su rendimiento:
  - La estrategia utilizada para dividir el trabajo de un procesador cuando éste recibe una petición de trabajo.
  - El esquema usado para determinar el procesador donador cuando un procesador entra en estado inactivo.