Algoritmos las Vegas

Universidad del Valle

750098M Simulación computacional

Contenido

1. Principios
2. Algoritmos tipo 1
3. Algoritmos tipo 2
4. Conclusiones

Principios

- Toman decisiones aleatorias para guiarse a una solución válida. Nunca dan una respuesta incorrecta.

- Algoritmos LasVegas tipo1 son los que siempre encuentran una solución (puede ser a costo de una eficiencia reducida). Son aleatorizaciones de algoritmos determinísticos.

Principios

- Algoritmos LasVegas tipo2 son los que no siempre encuentran una solución válida, pero admiten si es así. El algoritmo se puede repetir hasta encontrar una solución válida. (Dado que son algoritmos probabilísticos, repeticiones conducen a resultados diferentes)

Algoritmos tipo 1

- Se aplican en situaciones donde un algoritmo determinístico en su caso promedio tiene una eficiencia más alta que en el caso peor.
- La aleatorización convierte el caso peor al caso promedio.
- La ventaja del caso mejor se pierde, porque también se
- convierte en caso promedio.

Algoritmos tipo 1

Ejemplo QuickSort:

- **Selección del elemento pivote:**
  - El algoritmo deterministico: Primero de la lista
  - Algoritmo las Vegas: Elemento en posición aleatoria
- **Intercambio**: Los elementos mayores del pivote van a la derecha del pivote, los menores a su izquierda.
- **Recursión**: Se aplica Quicksort a las dos sublistas a la derecha y a la izquierda del pivote.

Algoritmos tipo 1

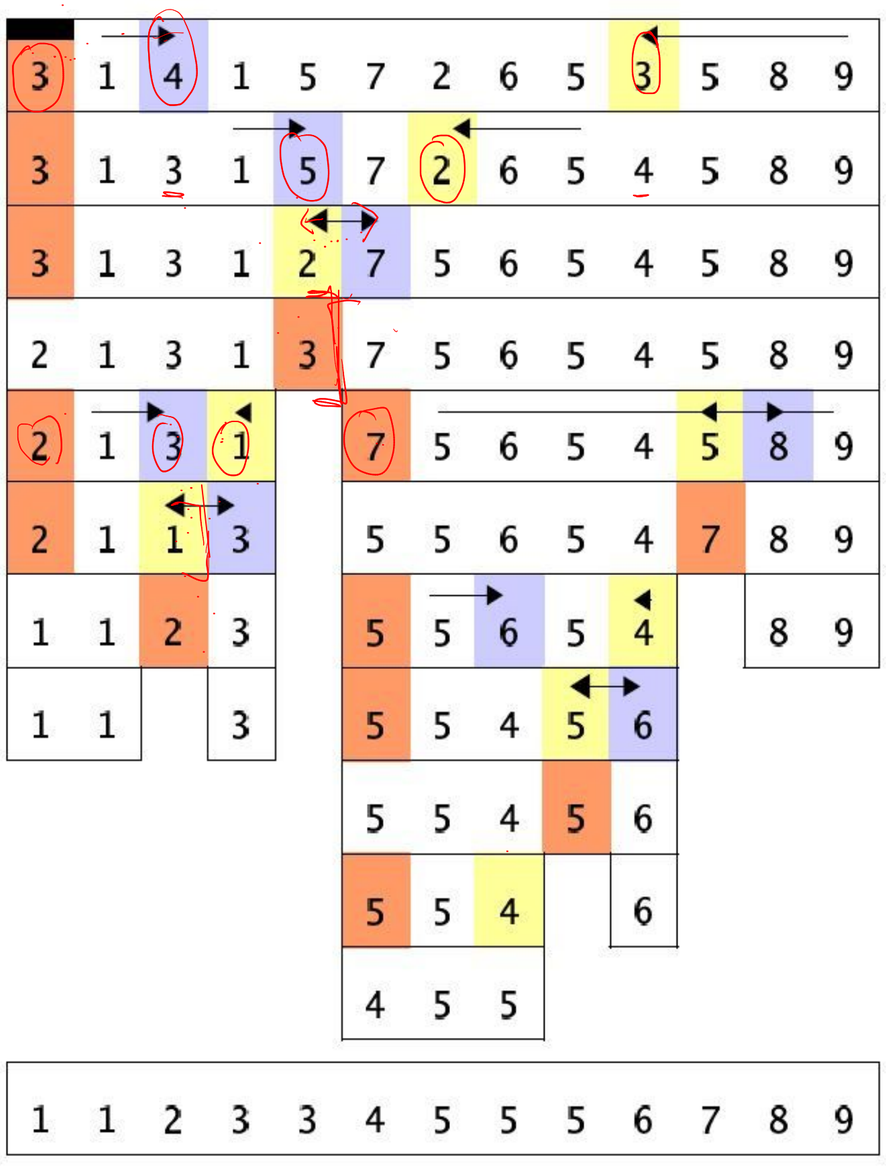
Ejemplo QuickSort:

- **Selección del elemento pivote:**
  - El algoritmo determinístico: Primero de la lista
  - Algoritmo las Vegas: Elemento en posición aleatoria
- **Intercambio**: Los elementos mayores del pivote van a la derecha del pivote, los menores a su izquierda.
- **Recursión**: Se aplica Quicksort a las dos sublistas a la derecha y a la izquierda del pivote.

Algoritmos tipo 1

Ejemplo QuickSort:

Ejemplo determinístico:



buscar desde la izquierda el primer elemento mayor que el pivote; desde la derecha el primero menor o igual al pivote;

intercambiar, reanudar la búsqueda;

intercambiar, reanudar la búsqueda;

dado que se cruzaron al búsquedas, intercambiar el pivote y el elemento final de la búsqueda desde la derecha.

aplicar el *quicksort* a los dos subvectores

En el mejor caso, se parte el vector cada vez en dos subvectores iguales

Algoritmos tipo 1

Ejemplo QuickSort:

Ejemplo determinístico (peor caso):

	2	3	4	5	6	7	8	9	10	11	12	13
1	2	3	4	5	6	7	8	9	10	11	12	13
1	2	3	4	5	6	7	8	9	10	11	12	13
1	2	3	4	5	6	7	8	9	10	11	12	13
1	2	3	4	5	6	7	8	9	10	11	12	13
1	2	3	4	5	6	7	8	9	10	11	12	13

..

1	2	3	4	5	6	7	8	9	10	11	12	13

buscar desde abajo el primer elemento mayor que el pivot, desde arriba el primero menor. si es igual, moverlo a lado del pivote

intercambiar, reanudar búsqueda; ya se han cruzado las búsquedas

...

En el peor caso, se separa cada vez un subvector de longitud 1

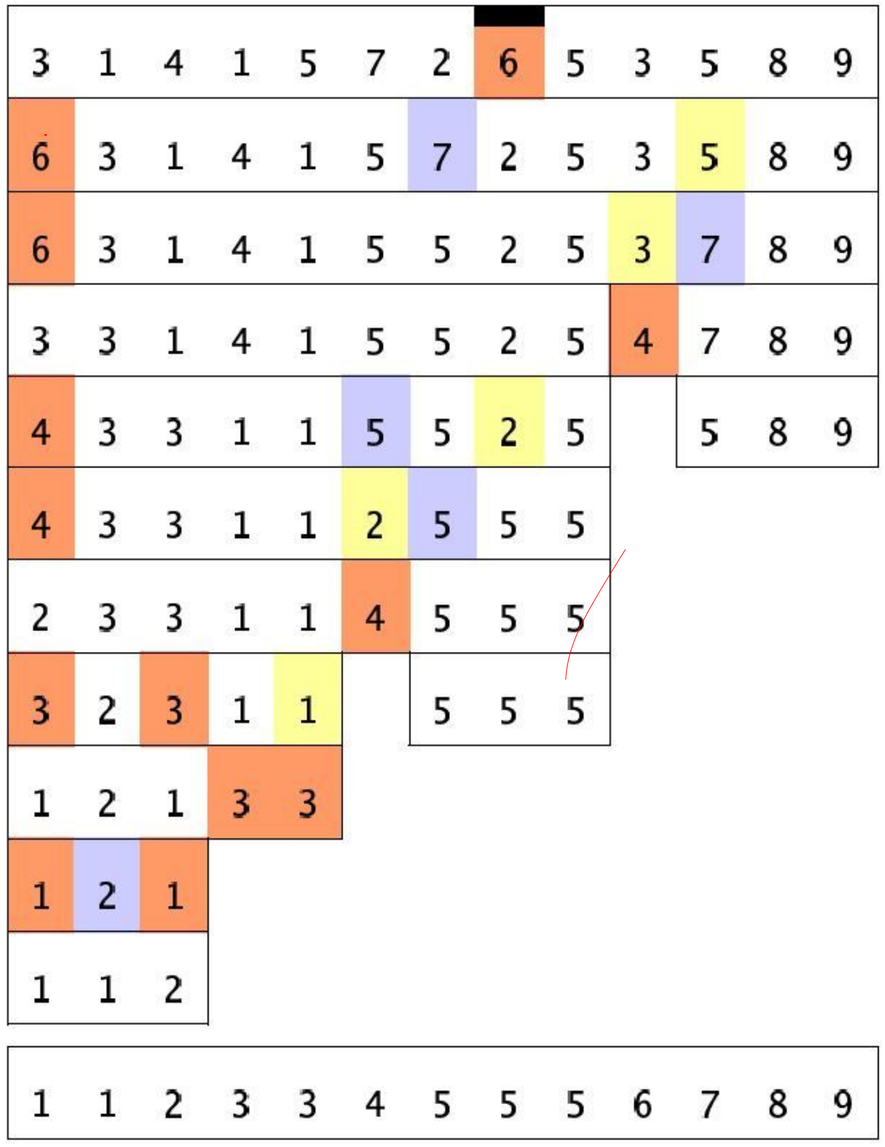
Algoritmos tipo 1

Ejemplo QuickSort:

Caso las Vegas

$aT(\frac{n}{b}) + f(n)$

$2T(\frac{n}{2}) + n \rightarrow O(n \log(n))$



indice aleatorio entre 1 y 13: 8

indice aleatorio entre 1 y 9: 4

indice aleatorio entre 1 y 5: 2

indice aleatorio entre 1 y 3: 1

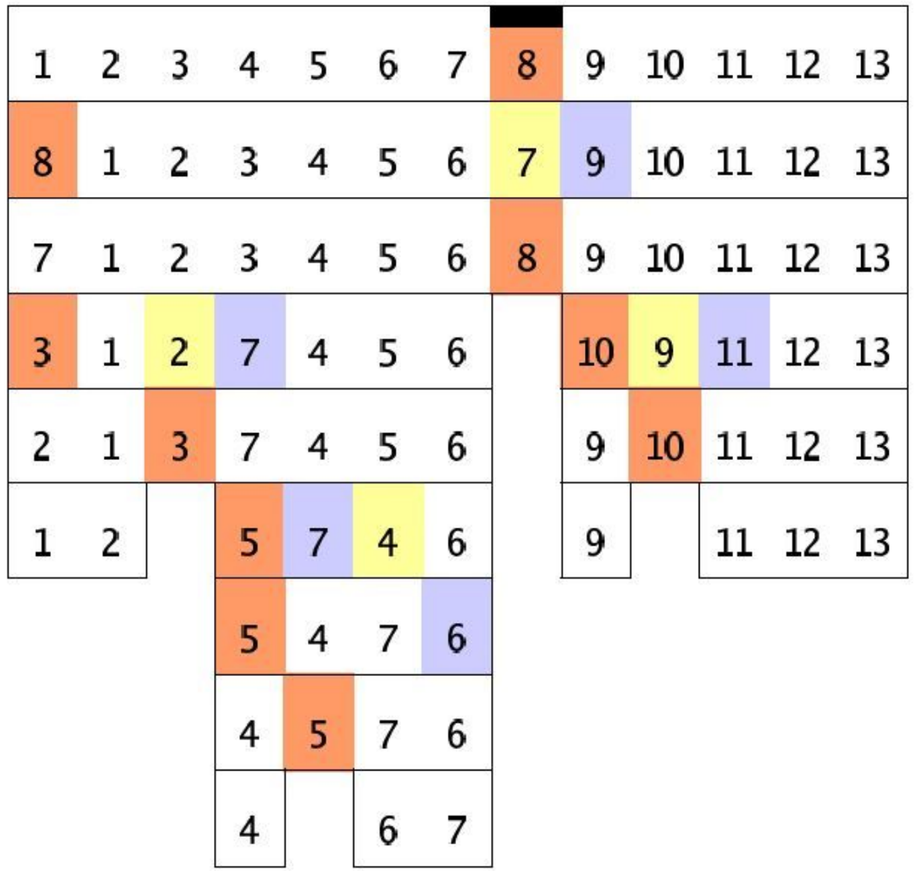
Seleccionar aleatoriamente el pivote, cada vez que se empieza un nuevo ciclo.
Es decir : un indice entre 1 y la longitud del vector en consideración.

Luego aplicar el quicksort de manera acostumbrada.

Algoritmos tipo 1

Ejemplo QuickSort:

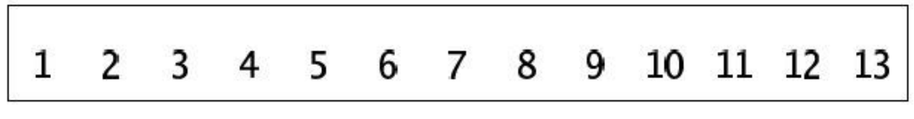
Caso las Vegas (Para el mejor caso del deterministico)



indice aleatorio entre 1 y 13: 8

indice aleatorio entre 1 y 7: 4; indice aleatorio entre 1 y 5: 2

indice aleatorio entre 1 y 4: 3



Por la selección aleatorizada del pivote se mejora mucho el caso peor del *quicksort*

Algoritmos tipo 1

Quicksort deterministico vs QuickSort (Las Vegas)

Deterministico	Aleatorizado
Funciona bien, si la lista está desorganizada. Eficiencia en el caso promedio: O(n logn)	La eficiencia esperada siempre corresponde al caso promedio: O(n logn)
Funciona muy mal, si la lista está casi ordenada. Eficiencia en el caso peor: $O(n^2)$	Resuelve más eficientemente los casos difíciles para el algoritmo determinístico. Pierde eficiencia en los casos fáciles para el algoritmo determinístico

Algoritmos tipo 1

La aleatorización mejora el caso peor y empeora el caso mejor.

Algoritmos tipo 2

Caracteristícas

- El algoritmo toma decisiones al azar para resolver el problema
- La toma de decisiones aleatorizadas puede conducir a una situación fracaso
- El algoritmo reconoce el fracaso
- En el caso de fracaso es suficiente volver a aplicar el algoritmo hasta encontrar una solución

Algoritmos tipo 2

Estructura de un algoritmo Las Vegas

```
algoritmo LasVegas(x, y, éxito)
    buscar solución a partir del input x
    si se encuentra solución,
            devolver   y ← solución
                       éxito ← verdadero
    sino    devolver   éxito ← falso
```

```
algoritmo RepetirLasVegas(x)
    repetir LasVegas(x, y, éxito)   hasta éxito
    devolver y
```

Algoritmos tipo 2

Estructura de un algoritmo Las Vegas

Para que funcione se debe saber que el algoritmo es capaz de encontrar la solución eventualmente.

Algoritmos tipo 2

Probabilidad de éxito.

Sea p(x) la probabilidad que el algoritmo LasVegas termine con éxito si se ejecuta con una entrada x.

- Se pide: $p(x) > 0$ para cada x. Esto asegura que para cada input x, el algoritmo encuentra una solución en algún momento.
- Más exigencia al algoritmo será pedir: Existe un $\delta > 0$ tal que $p(x) > \delta$ para cada posible input x (es decir, p(x) está acotado desde abajo)

Algoritmos tipo 2

Probabilidad de éxito.

Si solo se pide p(x) > 0, puede existir una secuencia de problemas siempre más complejos, podría nunca converger.

En el caso de que el algoritmo cumple la exigencia más fuerte, esta
situación no puede ocurrir.

Algoritmos tipo 2

Tiempo esperado las Vegas

El tiempo que se demora el algoritmo LasVegas es una variable aleatoria: Si los números aleatorios son favorables, se puede obtener una solución muy rápidamente, en otras ocasiones es algoritmo puede demorarse más. Buscamos el valor esperado de esta variable aleatoria.

Algoritmos tipo 2

Tiempo esperado las Vegas

Para una entrada x dada, sea

- **s(x)** es el tiempo de una aplicación del algoritmo LasVegas exitosa.
- **f(x)** es el tiempo de una aplicación del algoritmo LasVegas que fracasa

s(x) y f (x) son los dos valores que puede tomar LasVegas.

Algoritmos tipo 2

Tiempo esperado las Vegas

El valor esperado del tiempo de LasVegas para una entrada x E(x) entonces es:

$$E(x) = p(x) \cdot s(x) + (1 - p(x)) \cdot f(x)$$

tiempo exito

tiempo Fracaso

$p(x)$: Exito

$(1 - p(x))$: Fracaso

Algoritmos tipo 2

Tiempo esperado las Vegas

Sea t(x) el valor esperado (o su tiempo esperado) de RepetirLasVegas para la entrada x.

$$t(x) = \underbrace{p(x)s(x)}_{\text{exito en la primera vuelta}} + \underbrace{\big(1-p(x)\big)\big(f(x)+t(x)\big)}_{\text{fracaso en la primera vuelta y tiempo esperado en lo que sigue}}$$

Algoritmos tipo 2

Tiempo esperado las Vegas

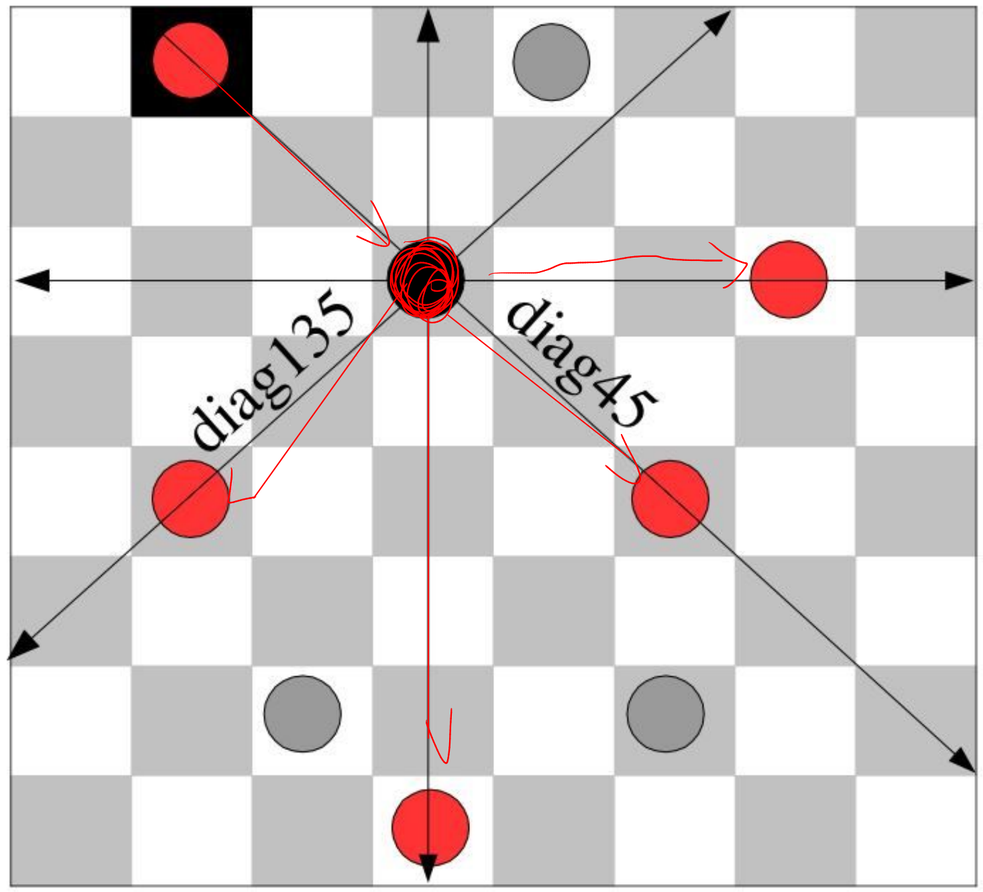
Esta recurrencia se resuelve fácilmente como:

$$t(x) = s(x) + \frac{1 - p(x)}{p(x)} f(x)$$

Algoritmos tipo 2

Ejemplo: Problema de n reinas

Colocar n reinas en un tablero de ajedrez de tal forma ninguna reina amenace a otra.



La reina negra amenaza las rojas, no amenaza las grises. Existe una posición para una reina que no amenaza a ninguna de las existentes?

Algoritmos tipo 2

Ejemplo: Problema de n reinas

Una reina amenaza a otra sí:

- se encuentra en la misma fila
- se encuentra en la misma columna
- se encuentra en el mismo diagonal de 45°
- se encuentra en el mismo diagonal de 135°

Algoritmos tipo 2

Ejemplo: Problema de n reinas

Ideas de solución: Se coloca una solo reina en cada fila. Esto reduce el problema a sólo seleccionar la columna

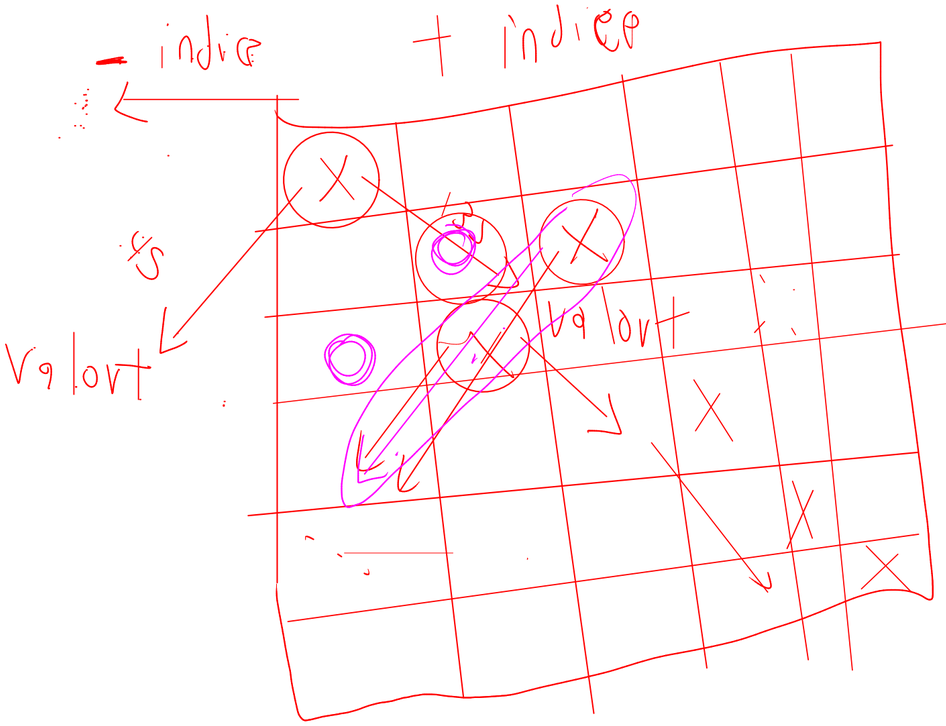
Se verifica si:

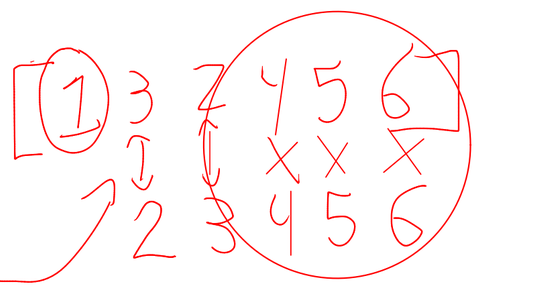
- Hay otra reina en la misma columna
- La posición de la reina no está en la diagonal (45° o 135°) de ninguna otra reina

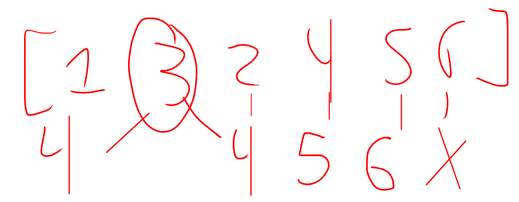
Algoritmos tipo 2

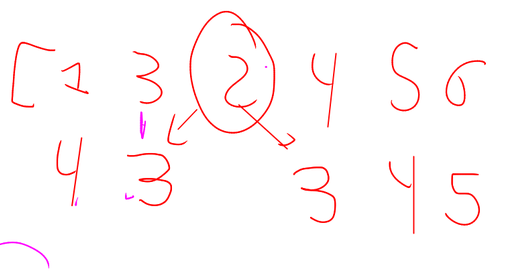
Ejemplo: Implementación

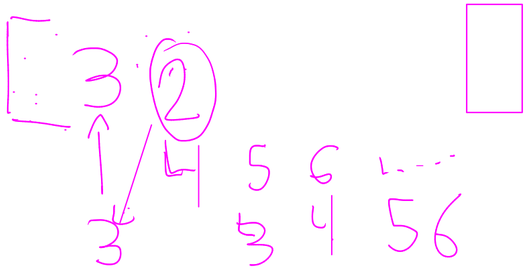
- Se crea un vector de tamaño n, cada posición representa una fila
- Se recorre el vector desde la primera posición hasta la última posición. Se selecciona un número aleatorio de un **conjunto entre 1 y n** que representa la columna (así evitamos que se amenacen entre columnas)
- Verificamos que no se amenacen por las diagonales

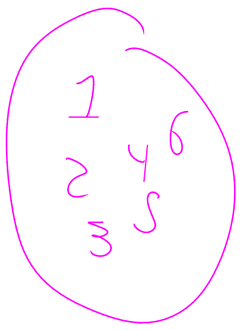

- indice
+ indice
Valor
Valor





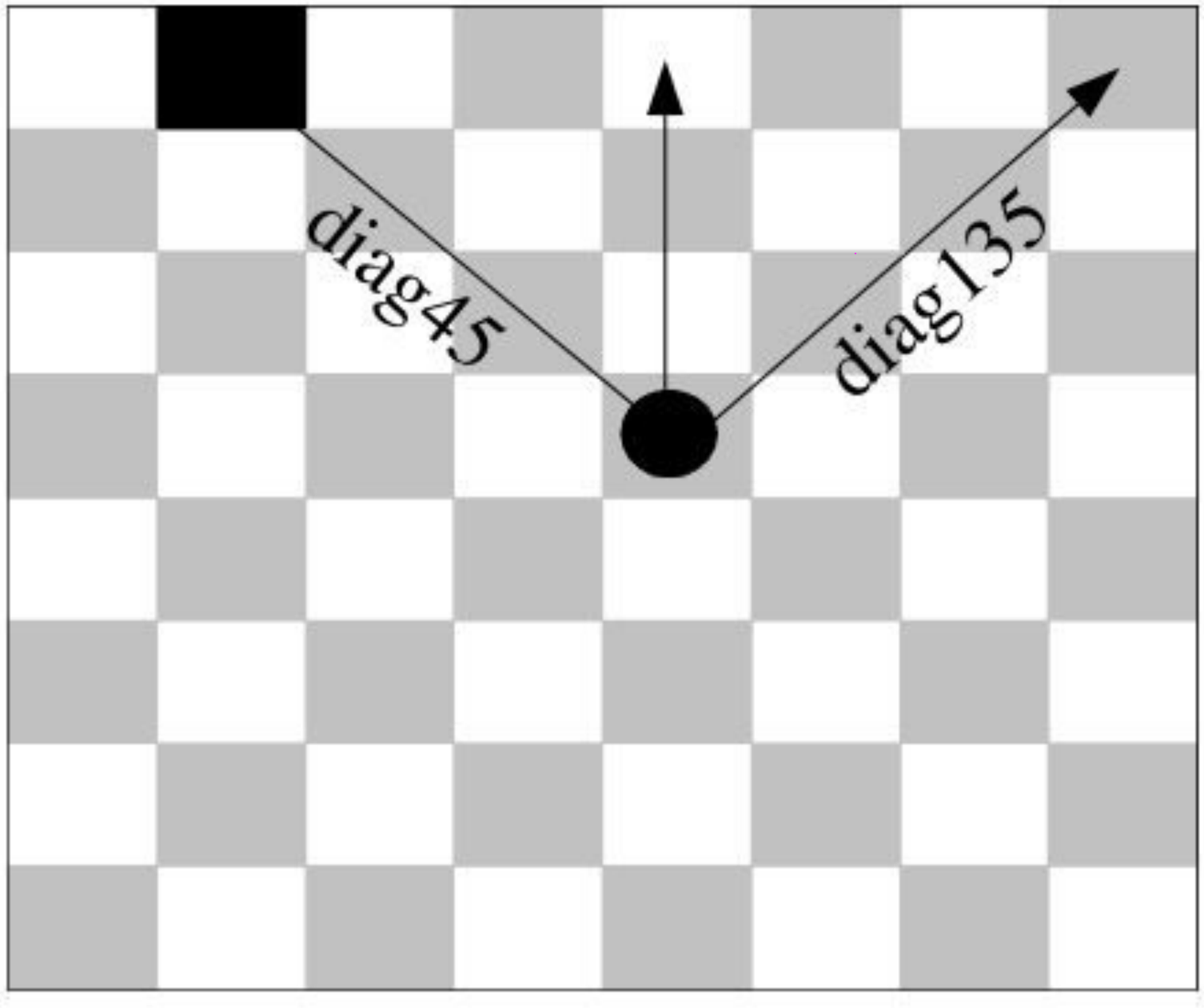






Algoritmos tipo 2

Ejemplo: Implementación



Algoritmos tipo 2

Ejemplo: Implementación

Para validar las diagonales, sólo es necesario realizarlo hacia arriba, ya que si hay una reina que nos amenace desde una posición inferior, esta validación se realizará con esta reina.

Algoritmos tipo 2

Ejemplo: Implementación

Si nuestra reina está en la fila i (posición i del vector) y columna j (valor de esa posición). Pasamos a verificar

- **Diagonal 45°:** Validar las posiciones anteriores del vector buscando los **antecesores** del valor almacenado en esa posición. Ejemplo

Algoritmos tipo 2

Ejemplo: Implementación

Diagonal 45°: Validar las posiciones anteriores del vector buscando los antecesores del valor almacenado en esa posición. Ejemplo, supongamos estamos en la posición 3 del vector

8	2	6	--	--

Validamos que en la posición 2 no esté el valor **5**, y en la posición 1 no esté el valor **4.**

Algoritmos tipo 2

Ejemplo: Implementación

Si nuestra reina está en la fila i (posición i del vector) y columna j (valor de esa posición). Pasamos a verificar

- **Diagonal 135°:** Validar las posiciones anteriores del vector buscando los **sucesores** del valor almacenado en esa posición. Ejemplo

Algoritmos tipo 2

Ejemplo: Implementación

Diagonal 135°: Validar las posiciones anteriores del vector buscando los sucesores del valor almacenado en esa posición. Ejemplo, supongamos estamos en la posición 3 del vector

8	2	6	--	--

Validamos que en la posición 2 no esté el valor **7**, y en la posición 1 no esté el valor **8.**

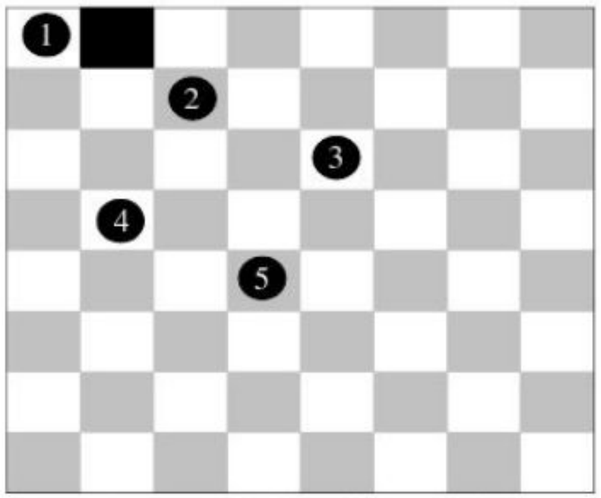
Algoritmos tipo 2

Ejemplo: Enfoque de solución

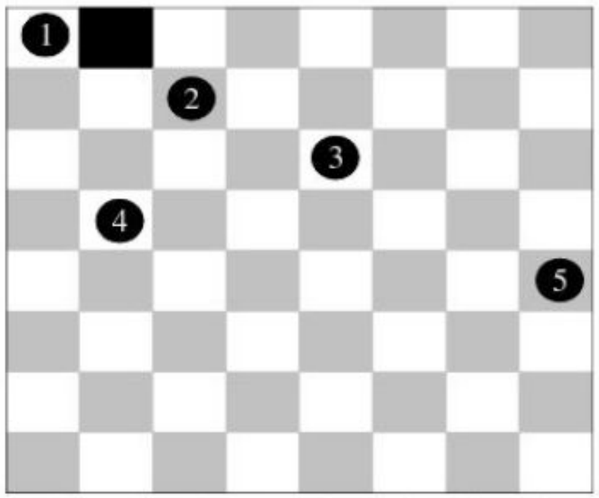
Deterministico	Las Vegas
Explorar sistemáticamente por un árbol de búsqueda las posiciones. Para cada fila: • Colocar una reina en una posición factible • Si no hay posición factible, regresar y seguir buscando	Colocar aleatoriamente las reinas. • Determinar una posición factible de una reina (escoger un elemento del conjunto) • Al escoger el elemento, determinar si es factible, si no hay elementos factibles, terminar sin éxtio.

Algoritmos tipo 2

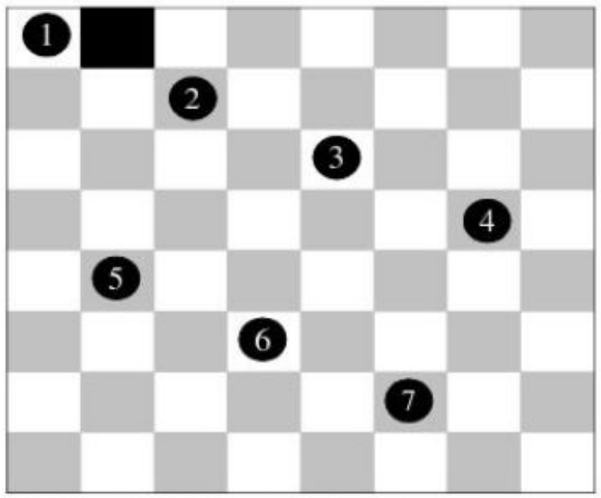
Ejemplo: Enfoque de solución determinístico:



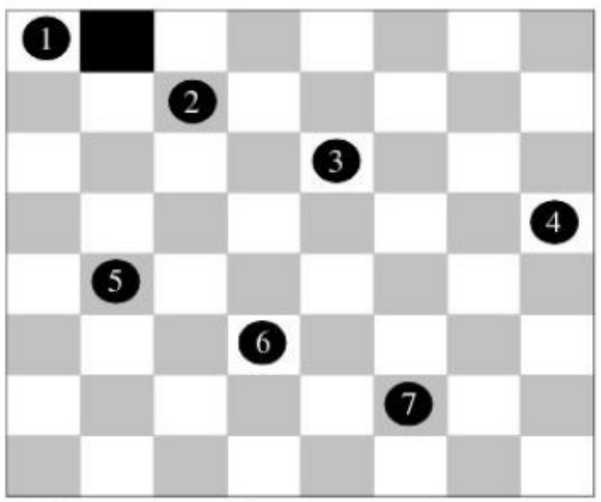
1. No hay posición disponible para reina 6, regresar un paso



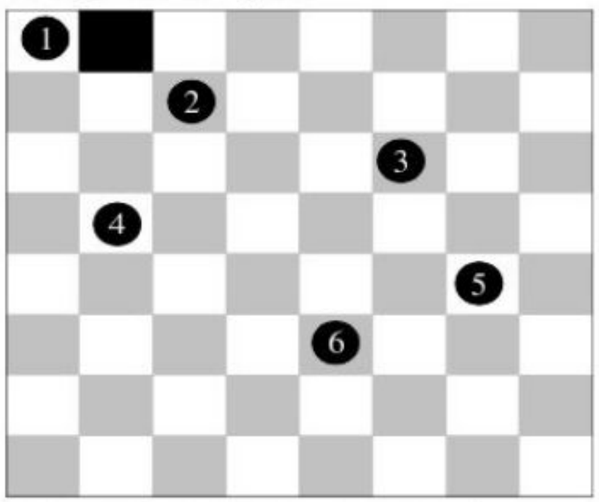
2. Cambio de posición de reina 5; no hay posición factible para reina 6, regresar un paso



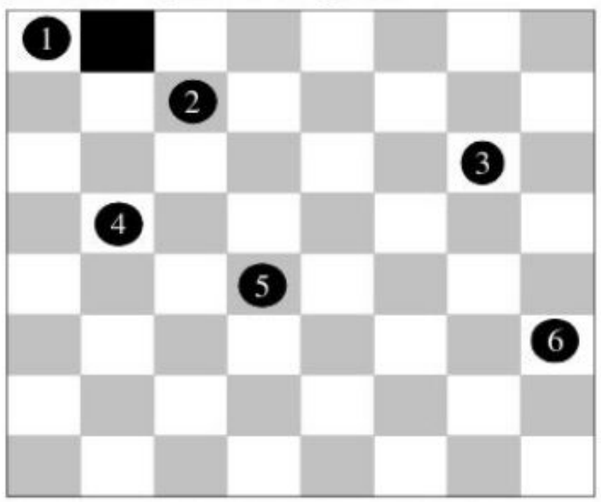
3. Cambio de posición de 4; colocar 5, 6, 7; no hay posición para 8; regresar un paso



4. Cambio de 4; colocar 5, 6, 7; no es posible colocar 8; regresar un paso



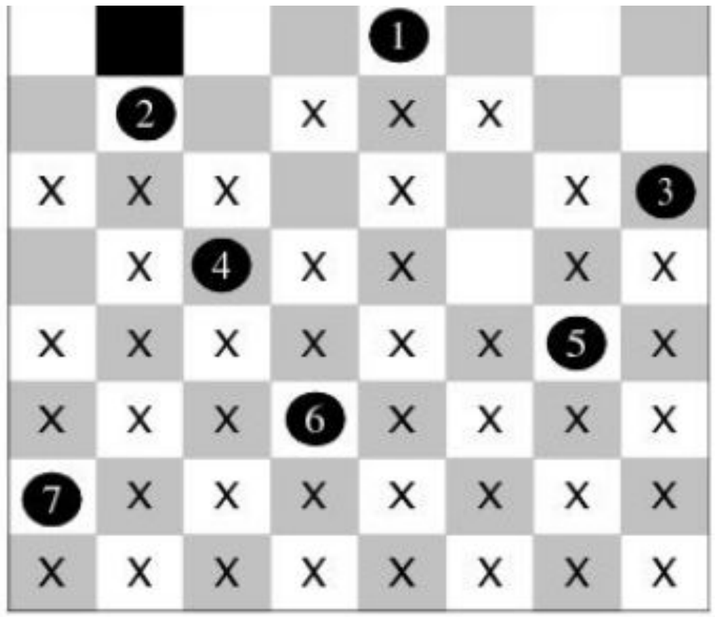
5. Cambio de 3; colocar 5, 6; no es posible colocar 7; regresar un paso



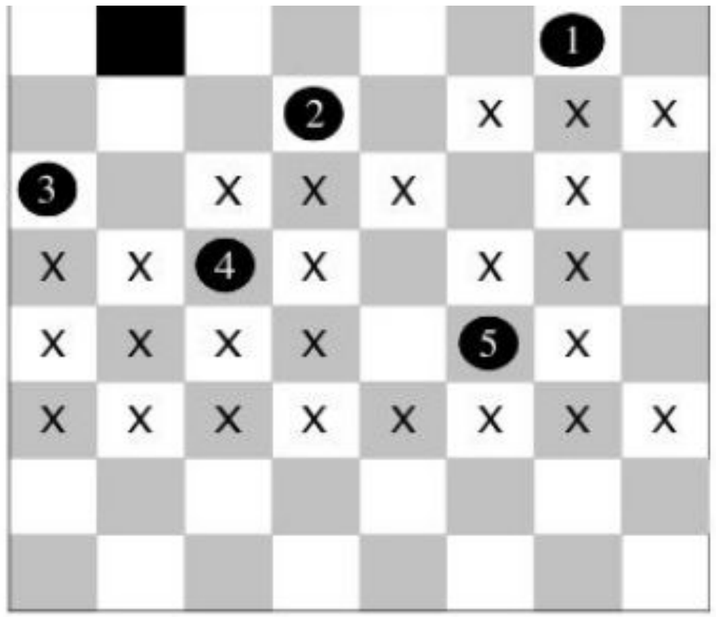
6. Cambio de 3; colocar 5, 6; no es posible colocar 7; regresar un paso ...

Algoritmos tipo 2

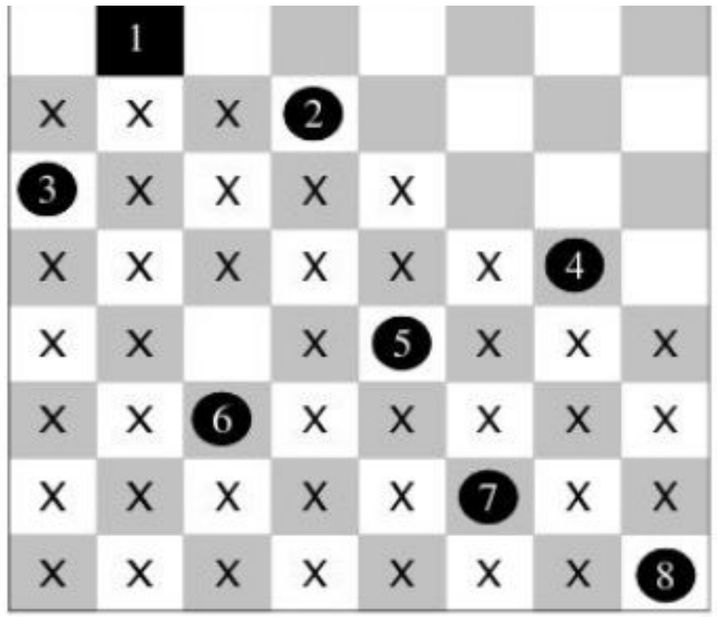
Ejemplo: Enfoque de solución las Vegas.



reina	posibilidades	aleatorio
1	8	3
2	5	4
3	3	3
4	3	2
5	1	3
6	1	1
7	1	1
8	0	-



reina	posibilidades	aleatorio
1	8	7
2	5	2
3	4	1
4	3	3
5	3	2
6	0	-
7	0	-
8	0	-



reina	posibilidades	aleatorio
1	8	2
2	5	1
3	4	1
4	2	1
5	2	2
6	1	1
7	1	1
8	1	1

Algoritmos tipo 2

Ejemplo: Análisis del algoritmo

- La exploración consiste en encontrar una solución factible ubicando 8 valores en 8 posiciones del vector..
- En caso de éxito se exploran en total 8 posiciones.
- A medida que se itera, se van colocando valores factibles, que indican las posiciones de las reinas
- Si en una posición dada no se puede colocar un valor, hemos fracaso.

Algoritmos tipo 2

Ejemplo: Analisis del algoritmo

- Experimentalmente se encuentra que la probabilidad de éxito es 0.1294
- El número esperado de posiciones que se logran llenar experimentalmente antes de una fracaso es 6.971

Algoritmos tipo 2

Ejemplo: Analisis del algoritmo

- Tomando:
  - s = 8. Número de posiciones exploradas en éxito.
  - p = 0.1294. Probabilidad de éxtio.
  - f = 6.971. Promedio de posiciones exploradas en fracaso.

$$t = s + \frac{1-p}{p} f = 54.90$$

Esto quiere decir que en promedio exploramos 55 posiciones. Es decir que necesitamos realizar al menos 8 ejecuciones en promedio para obtener un éxito.

Conclusiones

- Los dos tipos de algoritmos LasVegas toman decisiones aleatorias para conducirse a la solución de un problema.

- Los algoritmos LasVegas tipo 1 son las aleatorizaciones de algoritmos determinísticos. La aleatorización lleva cada problema al caso promedio.

- Los algoritmos LasVegas tipo 2 buscan la solución de un problema de manera aleatoria.

Conclusiones

Algoritmos las Vegas tipo 1

- La aleatorización lleva cada problema al caso promedio.

- Por eso son ventajosos siempre que la eficiencia del algoritmo determinístico en el caso promedio es menor que en el caso peor.

- En situaciones cercano al mejor caso, los algoritmos LasVegas empeoran la eficiencia.

Conclusiones

Algoritmos las Vegas tipo 2

- Si encuentran una solución, la devuelven; si no encuentran solución, lo reconocen y terminan sin éxito.
- Importante para la eficiencia del algoritmo es la probabilidad de tener éxito - se pide que sea diferente de 0 para cualquier entrada (sino, el algoritmo no sirve para solucionar el problema)
- Un algoritmo LasVegas se aplica iterativamente hasta encontrar la solución deseada.

Conclusiones

Algoritmos las Vegas tipo 2

- Para la eficiencia del algoritmo LasVegas y RepetirLasVegas son importantes: la probabilidad de éxito y el tiempo requerido en una situación de éxito y en una situación de fracaso
- Por consecuencia el diseño del algoritmo debe buscar el equilibrio entre la probabilidad de éxito (cuantas iteraciones se requiere) y los tiempos de ejecución: una probabilidad alta de éxito es buena, siempre y cuando no hace el algoritmo demasiado lento en una ejecución.