Técnicas y Paradigmas de Programación

Carlos Lagos

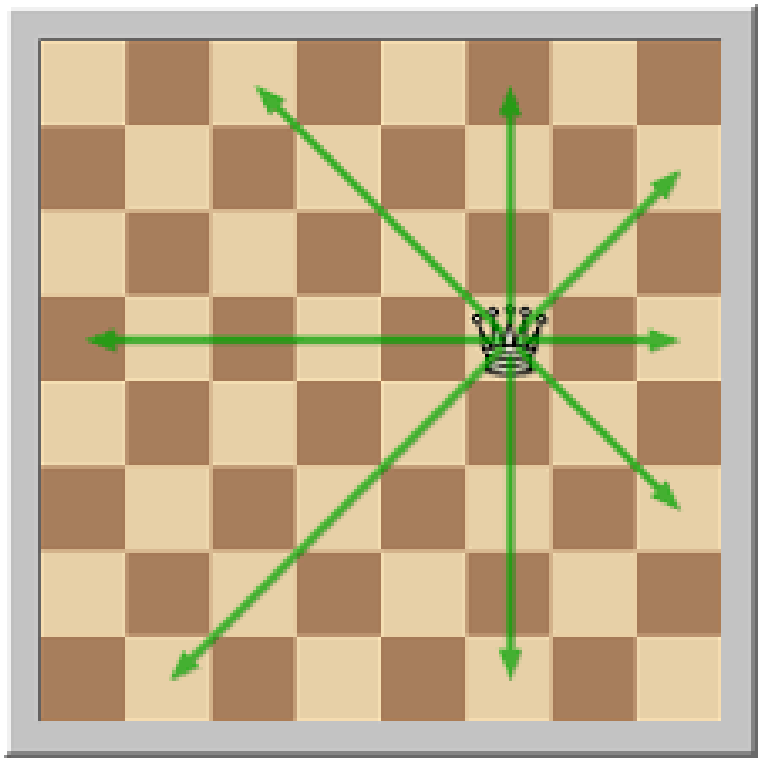
23 de julio de 2024

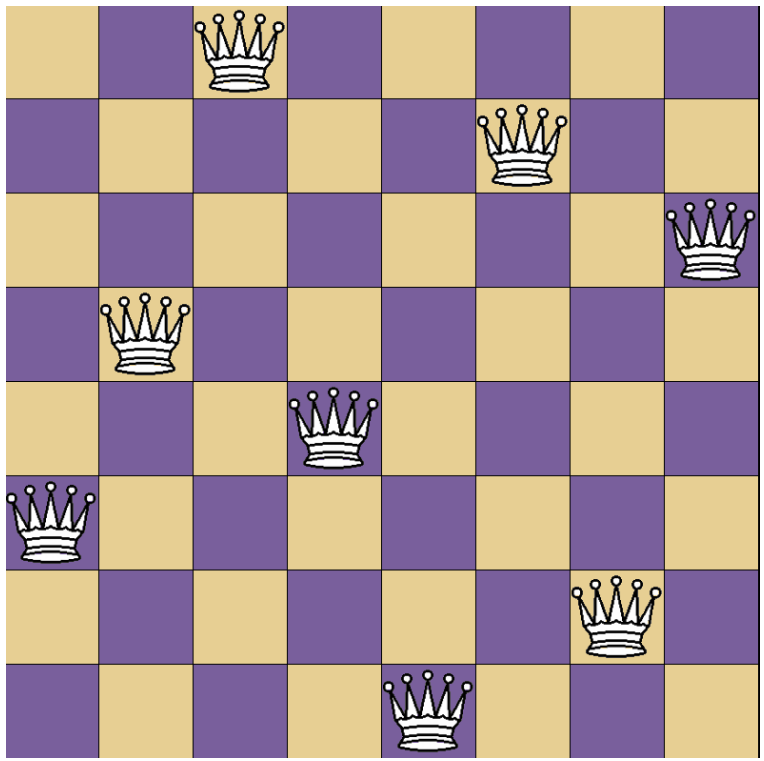
Contenidos

Problema de las Ocho Reinas

Enunciado

Consiste en colocar 8 reinas en un tablero de ajedrez de 8x8 de manera que ninguna pueda atacar a otra. Las reinas no deben compartir la misma fila, columna o diagonal. Encuentra todas las disposiciones que cumplan con esta condición.





Fuerza Bruta

Es una búsqueda exhaustiva de todos los posibles candidatos para las soluciones de un problema. Se itera a través de todos los estados posibles para encontrar una o varias soluciones óptimas y factibles.
Para solucionar el problema anterior, podemos usar 8 bucles 'for', uno por cada fila, iterando 8 veces en cada bucle para abarcar todas las columnas.

Fuerza Bruta - Ejemplo

Ejemplo con 4 reinas y un tablero de 4x4:

```
for(int i = 0; i < 4; i++) {
    for(int j = 0; j < 4; j++) {
        for(int k = 0; k < 4; k++) {
            for(int l = 0; l < 4; l++) {
                // Se verifica si es factible.
            }
        }
    }
}
```

Backtracking

Descripción

El backtracking es una técnica de búsqueda más eficiente que la fuerza bruta. Construye una solución paso a paso, verificando en cada etapa si sigue siendo factible. Si no es posible, retrocede y prueba otra opción, reduciendo así el espacio de búsqueda.

Backtracking

Descripción

El backtracking es una técnica de búsqueda más eficiente que la fuerza bruta. Construye una solución paso a paso, verificando en cada etapa si sigue siendo factible. Si no es posible, retrocede y prueba otra opción, reduciendo así el espacio de búsqueda.

En el problema de las 8 reinas, el backtracking coloca una reina en cada fila y verifica si la colocación es factible. Si una colocación resulta inviable, se retrocede y se prueba con otra opción, optimizando el proceso y evitando soluciones inviables.

Backtracking - Ejemplo

```
for (int i = 0; i < 4; i++) {
    if (!esFactible(i)) continue;
    for (int j = 0; j < 4; j++) {
        if (!esFactible(i, j)) continue;
        for (int k = 0; k < 4; k++) {
            if (!esFactible(i, j, k)) continue;
            for (int l = 0; l < 4; l++) {
                if (!esFactible(i, j, k, l)) continue;
                // Procesar soluci n factible
            }
        }
    }
}
```

También se puede implementar de manera recursiva.

Contenidos

Greedy

Supongamos que queremos resolver un problema P que tiene una solución óptima, a la que llamaremos R_o. Denotaremos por R_g una solución construida mediante una estrategia codiciosa (greedy).
Una estrategia Greedy consiste en construir una solución eligiendo los óptimos locales en cada paso, según los criterios establecidos por la estrategia.

Greedy - Problema de las Monedas

Enunciado

Supongamos que eres millonario y necesitas comprar pan, cuyo costo es de 13 unidades de una moneda imaginaria. En tu país, solo se utilizan monedas con los siguientes valores: $\{1, 6, 7, 10\}$. Para minimizar el número de monedas que usas, debes encontrar la forma de pagar el pan con la menor cantidad posible de monedas.

Greedy - Problema de las Monedas

Podríamos usar una estrategia greedy en la que, en cada paso, seleccionemos la moneda que más reduce la cantidad restante por pagar.

Greedy - Problema de las Monedas

Podríamos usar una estrategia greedy en la que, en cada paso, seleccionemos la moneda que más reduce la cantidad restante por pagar.

- ¿Es esta estrategia óptima? ¿$R_g = R_o$?

Greedy - Problema de las Monedas

Podríamos usar una estrategia greedy en la que, en cada paso, seleccionemos la moneda que más reduce la cantidad restante por pagar.

- ¿Es esta estrategia óptima? ¿$R_g = R_o$?
- Si no es óptima, ¿qué utilidad tiene?

Greedy - Problema de las Monedas

Podríamos usar una estrategia greedy en la que, en cada paso, seleccionemos la moneda que más reduce la cantidad restante por pagar.

- ¿Es esta estrategia óptima? ¿$R_g = R_o$?
- Si no es óptima, ¿qué utilidad tiene?
- ¿Existen problemas en los que la solución greedy R_g coincide con la solución óptima R_o?

Greedy en Programación Competitiva

En programación competitiva, dado que las soluciones deben resolver el problema de manera exacta, buscamos que la solución greedy R_g sea igual a la solución óptima R_o. Esto se cumple para ciertos problemas, dependiendo de la estrategia utilizada. Para garantizar esto, es necesario demostrar que la estrategia greedy es óptima.

Greedy - Diferencia Máxima entre Subconjuntos

Enunciado

Dado un arreglo de enteros no negativos $A[1..n]$ y un entero k $(0 \leq k \leq n)$, encuentra dos subconjuntos de A de tamaños nk y k tal que la diferencia entre sus sumas sea máxima.

Greedy - Diferencia Máxima entre Subconjuntos

```
int main(){
    int n, k;
    cin >> n >> k;

    vector<int> A(n);
    int suma_total = 0, suma_menores = 0;
    for(int i = 0; i < n; i++){
        cin >> A[i];
        suma_total += A[i];
    }
    sort(A.begin(), A.end());
    int menores = min(k, n - k);
    for(int i = 0; i < menores; i++) suma_menores += A[i];
    int res = (suma_total - suma_menores) - suma_menores;
    cout << res << endl;
    return 0;
}
```

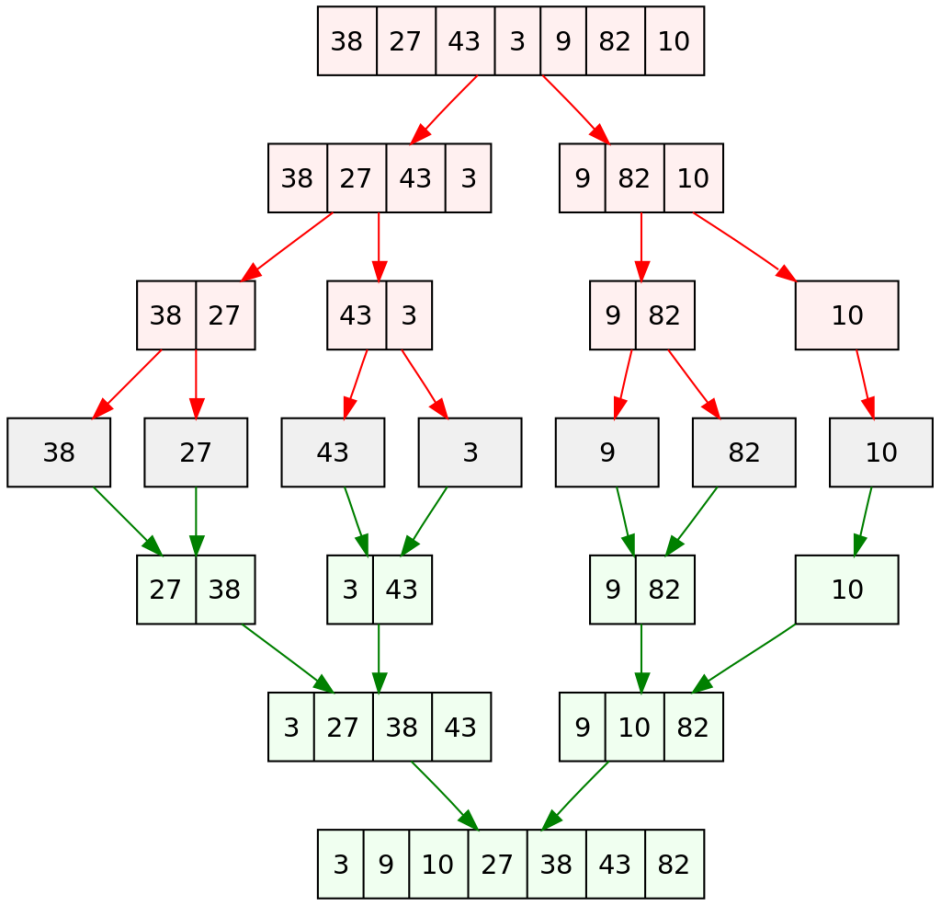
Contenidos

Dividir y Conquistar

Este enfoque consiste en dividir un problema en partes más pequeñas, resolver cada parte por separado y luego combinar las soluciones parciales para resolver el problema original. Algoritmos como mergesort y quicksort utilizan este enfoque.

MergeSort

Este algoritmo de ordenamiento aprovecha que unir dos arreglos ordenados para producir otro arreglo también ordenado tiene una complejidad de O(n + m).



Contenidos

Búsqueda binaria

Descripción

Esta técnica utiliza un enfoque llamado Decrecer y Conquistar, donde en cada paso se busca reducir el problema.
Para aplicar la búsqueda binaria, es necesario que aquello a lo que se le aplique tenga un comportamiento monótono, lo que nos permite descartar la mitad de las posibilidades en cada paso.

Búsqueda binaria

Descripción

Esta técnica utiliza un enfoque llamado Decrecer y Conquistar, donde en cada paso se busca reducir el problema.
Para aplicar la búsqueda binaria, es necesario que aquello a lo que se le aplique tenga un comportamiento monótono, lo que nos permite descartar la mitad de las posibilidades en cada paso.

Función monótona

Una función f es monótona si $x \leq y$ implica $f(x) \leq f(y)$ (creciente) o $f(x) \geq f(y)$ (decreciente).

Búsqueda de un elemento en un arreglo ordenado

Enunciado

Dado un arreglo $A[1...n]$ ordenado de forma no decreciente, se quiere verificar si el elemento x está presente en este arreglo.

Búsqueda de un elemento en un arreglo ordenado

Enunciado

Dado un arreglo $A[1...n]$ ordenado de forma no decreciente, se quiere verificar si el elemento x está presente en este arreglo.

Observaciones

- Si recorremos cada elemento de izquierda a derecha, la complejidad sería $O(n)$.

Búsqueda de un elemento en un arreglo ordenado

Enunciado

Dado un arreglo $A[1...n]$ ordenado de forma no decreciente, se quiere verificar si el elemento x está presente en este arreglo.

Observaciones

- Si recorremos cada elemento de izquierda a derecha, la complejidad sería $O(n)$.
- ¿Se puede mejorar esto usando búsqueda binaria?

Búsqueda de un elemento en un arreglo ordenado

Enunciado

Dado un arreglo $A[1...n]$ ordenado de forma no decreciente, se quiere verificar si el elemento x está presente en este arreglo.

Observaciones

- Si recorremos cada elemento de izquierda a derecha, la complejidad sería $O(n)$.
- ¿Se puede mejorar esto usando búsqueda binaria?
- ¿Qué complejidad tendría?

Factory Machines

Enunciado

Una fábrica tiene n máquinas que se pueden usar para fabricar productos. Tu objetivo es producir un total de t productos. Para cada máquina, conoces el número de segundos que necesita para hacer un solo producto. Las máquinas pueden trabajar simultáneamente, y puedes decidir libremente su horario. ¿Cuál es el tiempo más corto necesario para fabricar t productos?

Contenidos

Dos punteros

La técnica de dos punteros se utiliza para encontrar dos posiciones en un arreglo que cumplan con una condición específica. Consiste en usar dos punteros para indicar dos posiciones distintas en el arreglo y mover estos punteros según un criterio determinado.

Suma de dos valores

Enunciado

Se te da un arreglo de n enteros, y tu tarea es encontrar dos valores (en posiciones distintas) cuya suma sea x.

Ventana Deslizante

La técnica de ventana deslizante (Sliding window) se utiliza para encontrar subarreglos o subsecuencias que cumplan con una condición específica. Consiste en mantener una ventana que se mueve a lo largo del arreglo, ajustando su tamaño y posición según un criterio determinado. A medida que la ventana se desplaza, se evalúan las subsecuencias dentro de ella para cumplir con la condición requerida.

Playlist

Enunciado

Se te da una lista de reproducción de una estación de radio desde su establecimiento. La lista de reproducción tiene un total de n canciones. ¿Cuál es la secuencia más larga de canciones sucesivas en la que cada canción es única?