Taller de Programación

www.inf.ucv.cl/ wpalma/taller

Dr. Wenceslao Palma
wenceslao.palma@ucv.cl

Backtracking

- Backtracking se asemeja a un recorrido en profundidad dentro de un grafo dirigido.
- El grafo en cuestión suele ser un árbol y siempre existe en forma implícita.
- El objetivo del recorrido es encontrar soluciones para algún problema.
- La solución se consigue construyendo soluciones parciales a medida que progresa el recorrido. Una solución parcial limita las regiones en las que se puede encontrar una solución completa.
- El recorrido es exitoso si siguiendo este procedimiento es posible definir por completo una solución.

Un recorrido no tiene éxito si en alguna etapa la solución parcial, obtenida hasta el momento, no se puede completar. En este caso, el recorrido vuelve hacia 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, selecciona uno y prosigue el recorrido en busca de una solución.

Backtracking

En términos más formales:

- Se busca una solución en forma de secuencia $X_1, X_2, X_3, ..., X_n$ tal que cumpla una restricción $R(X_1, X_2, X_3, ..., X_n)$.
- La restricción se puede plantear considerando restricciones $R^{(1)}(X_1)$,$R^{(2)}(X_1, X_2)$,...,$R^{(n)}(X_1, X_2, ..., X_n) = R(X_1, X_2, X_3, ..., X_n)$. Donde básicamente $R^i(.)$ significa "hasta aquí cumple".
- El encontrar la solución implica recursividad.

Backtracking es una técnica muy general la cual debe ser adecuada para cada problema en particular. Además, es muy importante la forma en la cual representar una solución.

Backtracking: el problema de las n-reinas

- el enfoque naive $\rightarrow$ TLE :(
- Se sabe que dos reinas no pueden estar en la misma columna. Luego se puede simplificar el problema a encontrar una permutación de $8! = 40k$ posiciones de fila.
- Por ejemplo row=$\{2,4,6,3,1,7,5\}$ es una solución donde row[1]=2 significa que la reina de la columna 1 se encuentra en la fila 2.
- Además, dos reinas no pueden estar en la misma diagonal. Si tenemos una reina en la posición (i, j) y otra en la posición (k, l), una ataca a la otra si $abs(i - k) == abs(j - l)$

Backtracking: el problema de las n-reinas[1]

```
 void NQueens(int queen) {
     int row,j;
     for (row = 1; row <= 8; row++)
          if (place(queen, row)) { // if can place this queen at this row?
             x[queen] = row; // put this queen in this row
             if (queen == 8 && x[b] == a) { // a candidate solution & (a, b) has 1 queen
                printf("%2d %d", ++lineCounter, x[1]);
                for(j = 2; j <= 8; j++) printf(" %d", x[j]);
                printf("\n");
             }
             else
                NQueens(queen + 1); // recursively try next position
          }
}

int place(int queen, int row) {
     int prev;
     for(prev = 1; prev <= queen - 1; prev++) // check previously placed queens
       if (x[prev] == row || (abs(x[prev] - row) == abs(prev - queen)))
          return 0; // an infeasible solution if share same row or same diagonal
     return 1;
}
```

[1] código extraído desde Competitive Programming, 1ra Ed.