CS F364
Design & Analysis of Algorithms

Algorithms – Design Techniques

Exact Solutions - Search with Backtracking: Template

BACKTRACKING - APPROACH

- Solution space can also be viewed as "certificate" space:
  - i.e. Searching for a solution can be viewed as "constructing" a "valid" certificate (and testing it)
- Recall non-deterministic (ND) machines
  - ND machines can be simulated deterministically by exploring all possibilities exhaustively !
- Template for "backtracking"
  1. Systematically construct solutions and test them
     - If a test fails backtrack to find an alternate solution
  2. Repeat step 1 until valid solution is found.

BACKTRACKING - ALGORITHMIC TEMPLATE - OUTLINE

```
Algorithm_Template Backtrack(x):
 // x is a problem instance

    F = { (x,{}) }          // F is a set of configurations
    while (F not empty) do {
       // inspect configurations in F one by one




    }
    return “no solution”
```

BACKTRACKING - ALGORITHMIC TEMPLATE - CONFIGURATIONS

```
Algorithm_Template Backtrack(x):  // x is a problem instance
    F = { (x,{}) }          // F is a set of configurations
    while (F not empty) do {
       // inspect configurations in F one by one
       select the most promising configuration (x,y) from F;
       expand (x,y) by making additional choices to get a set of
            new configurations C = { (x1,y1), (x2,y2), …,(xk,yk) };
        for each (xj,yj)  in C {
           // validate(xj,yj)


        }
    }
    return “no solution”
```

BACKTRACKING - ALGORITHMIC TEMPLATE

```
Algorithm_Template Backtrack(x):  // x is a problem instance
    F = { (x,{}) }          // F is a set of configurations
    while (F not empty) do {
      // inspect configurations in F one by one
      select the most promising configuration (x,y) from F;
      expand (x,y) by making additional choices to get a set of
            new configurations C = { (x1,y1), (x2,y2), …,(xk,yk) };
       for each (xj,yj)  in C {
          // validate(xj,yj)
           if "solution found" return the solution derived from (xj,yj);
           else if  "dead end" then discard;  // backtrack
           else F = F U { (xj,yj) }
       }
    }
    return "no solution"
```