Complete Search

Sometimes you just have to check every last thing

Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

December 31, 1979

Complete Search

Sometimes there is no clever way to find the answer, you just have to check all the combinations until you find what you are looking for. There are often ways to do this efficiently though!

Objectives

- Indentify the characteristics of a problem that requires complete search.
- Explain how pruning and representations can help make our search more efficient.
- Review some common examples of complete search.

Overview

- Sometimes the only way to solve a problem is to iterate over the entire search space.
  - Reputation is that these are “easy”, but this is not necessarily the case.
- Two major classes:
  - Iterative
  - Recursive Descent

Interative Problems

- Typically a `for` loop or many nested `for` loops.
    - You can sometimes simplify the search range if you think about the problem.
    - You can also speed things up by being clever with your testing.
- Two patterns if you want to generate a sequence of numbers that have property P:
    - Brute-force generate all of the sequences that have property P.
    - Generate all sequences and check which ones have property P.
- You will need to do the math to see which is most feasable.

Example: UVa 11742 - Social Constraints

Example taken from CP4

Suppose there up up to 8 movie-goers. There are a set of $0 \leq m \leq 20$ constraints of the form "person *a* must be within *x* seats of person *b*", or "person *a* must be at least *x* seats away from person *b*." How would you do this?

Example: UVa 11742 - Social Constraints

Example taken from CP4

Suppose there up up to 8 movie-goers. There are a set of $0 \leq m \leq 20$ constraints of the form "person *a* must be within *x* seats of person *b*", or "person *a* must be at least *x* seats away from person *b*." How would you do this?

Technique

- Generating the sets that satisfy the constraints directly will be tricky.
- Instead, iterate all the seatings and check which ones satisfy all the constraints!
- C++ has a 'next$_{\text{permutation}}$' function that can help with this.

Example: UVa 12455 - Bars

Example taken from CP4

Suppose you have a list of 20 numbers. Can you find a subset of those numbers that sum to X?

Example: UVa 12455 - Bars

Example taken from CP4

Suppose you have a list of 20 numbers. Can you find a subset of those numbers that sum to X?

Technique

- Use a bitmask to represent which items are selected.

```
vi l = ... ; // numbers go in here
int x;

for(i=0; i< 2^20; ++i) {
  int sum = 0;
  for(int j = 0; j<20; ++j)
     if (i & (1<<j)) // see if bit 'j' is on?
       sum += l[j]; // if yes, process 'j'
  if (sum == X) break;
}
```

Recusive Descent Problems

N-Queens

Josephus Problem