

S7: Search
CSci 2041:
Advanced Programming Principles

University of Minnesota,
Prof. Van Wyk,
Spring 2018

1

Search

Many problems can be seen as search problems.

These involve an exploration of some **search space**.

- ▶ How is this space specified?
- ▶ What are the elements in the space?
- ▶ How are they related?
- ▶ Does one transition from one state in the search space to another?

2

For example, logical tautologies

- ▶ A **tautology** is a formula that is always true, for any values of its free variables.
- ▶ Is the formula $P \vee \neg P$ a tautology?
- ▶ Is the formula $(P \rightarrow Q) \vee (Q \rightarrow P)$ a tautology?
- ▶ That is, is the formula always *true* for any Boolean values assigned to P and Q ?
- ▶ Search for a counter example in all assignments. If we don't find one, then the formula is a tautology.

3

Another example, subset sum

- ▶ Given a set of integers (positive and negative), is there a non-empty subset that sums up to 0?
- ▶ Consider $\{1, 3, -2, 5, -6\}$.
The subset $\{1, 5, -6\}$ sums to 0.
- ▶ Search all possible subsets for one that sums to 0.
- ▶ How do we generate all of these?

4

Wolf-Goat-Cabbage

Consider the problem of a person needing to move their wolf, goat, and cabbage across a river in his canoe, under the following restrictions:

- ▶ The canoe holds only the person and one of the wolf, goat, or cabbage.
- ▶ The goat and cabbage cannot be left unattended or the goat will eat the cabbage.
- ▶ The wolf and the goat cannot be left unattended or the wolf will eat the goat.
- ▶ Only the person can operate the canoe.

Is there a sequence of moves in which the man can safely transport all across the river with nothing being eaten?

5

Search techniques

Three questions:

1. How do we enumerate the elements of the search space?
2. How we proceed from one to the next?
3. How do we stop when we've found one?

6

Enumerating the search space

- ▶ This is rather problem specific.
- ▶ We can often think of this as a tree or graph exploration problem.
- ▶ We'll focus on representing the search space as a tree first.
- ▶ How might we visualize the search space for subset-sum? Let's draw some trees on the white board.

7

- ▶ We may not want to generate the entire search space.
- ▶ Maybe just find the first,
- ▶ or maybe just the first few until a result is satisfactory.
- ▶ We can use `options`, `exceptions`, and `continuations` to control our searching process.
- ▶ We see each of these in turn below.

9

Using options

- ▶ We may not want to generate all possible states.
- ▶ Maybe we find what we want without looking at them all.
- ▶ Or maybe we want to look at a few before accepting a solution as acceptable.
- ▶ Can we use an OCaml `option` to indicate that we've found what we are looking for?
- ▶ Consider the code in `search_options.ml` in the code-examples directory of the public repository.

Using exceptions

- ▶ Another way to change the flow of control is to raise an exception.
- ▶ We could do this when we find what we are looking for, or when we have failed and need to continue.
- ▶ Again, see the samples in [search_exceptions.ml](#).

11

Continuations

- ▶ Continuation passing style (CPS) is a style of writing programs in which the computation that happens after a function returns is packaged as a function and passed to that function instead where it is called directly.
- ▶ In CPS, functions do not return. The computation that happens next is passed along as an argument in the form of a continuation function.
- ▶ In the subsetsum problem we pass two continuations, one to evaluate if we succeed and find a subset that sums to 0, and another one in the case in which we fail and reach a dead-end in the search process.
- ▶ See the code in [subsetsum.cps.ml](#) in the code-examples directory of the public repository for more examples and discussion.

12

Searching graphs

- ▶ Let's consider the wolf-goat-cabbage problem again.
- ▶ We must now deal with state search that resembles a graph instead of a tree.
- ▶ That is, our state search may bring us back to states we've already visited.
- ▶ Let's consider the search space as a graph on the white board.
- ▶ Then, we'll develop a solution that can be found in [wolf.ml](#) in the code examples repository.

13