Advanced Topics: Bipartite Matching

Imagine you have two items that have to be assigned together:
- FPGA I/O signals and pins that can support specific protocols
- Men and women for arranged marriages

Graph form:
- Node sets A and B, and edges V from A to B representing compatibility

Possible goals:
1.) Find a perfect matching of elements from A to B, such that pairs are compatible.
2.) Find a maximum matching if a perfect matching not possible (more A’s than B’s? Not right compatibilities?).

Active Learning: Matchmaker

Find the most possible marriages that can be formed (no polygamy, etc).
**Bipartite Matching Algorithm – Iterative Improvement**

Maze routing/BFS to find alternating path from unmatched A to unmatched B

Alternating path: Goes A->B->A->B ... Begins in A, ends in B

A->B edges not in matching, B->A edges already in matching

Flip status of edges on alternating path (in matching <-> not in matching)

---

**Bipartite Matching Example**

---
Weighted Matching

Same as bipartite matching, but edges have score – want matching with highest total edge weight
FPGA I/O assignment – power savings?
Marriage – bribes from thankful parents?

Weighted Matching Algorithm

Iteratively find largest benefit alternating path.
Route from unmatched-A to unmatched-B (Women to Men)
All edges already in matching have normal cost, direction from bottom to top
All other edges have negative cost, direction from top to bottom
Advanced Topics: Dynamic Programming

Dynamic Programming:
Solve problem by breaking them into problems, combine the sub-solutions
If some subproblems repeat, save the solutions to reuse instead of recalc.

Example: Fibonacci  \( \text{Fib}(N) = \text{Fib}(N-1) + \text{Fib}(N-2), \text{Fib}(1) = 0, \text{Fib}(2) = 1 \)

Dynamic Programming solution:

```c
Fib(N) {
    if not(saved(N))
        saved(n) = Fib(N-1) + Fib(N-2);
        return saved(N);
    }
```

Other examples from 541 of dynamic programming?

Dynamic Programming Requirements

Optimal Substructure: optimal solution can be found by combining optimal
solutions to subproblems.

Overlapping Subproblems: space of subproblems is small
recursive algorithm will solve the same subproblems over and over

Note: if subdivision reduces problem size significantly (i.e. halves), then it is
referred to as "divide-and-conquer". Merge-sort, etc.
Dynamic Programming Example

Map a circuit into 4-LUTs with the fastest design (fewest LUT levels in->out)
Assume input is already subdivided into <=4 input gates.