You think you know when you can learn, are more sure when you can write, even more when you can teach, but certain when you can program.
Combinatorial Optimization
Types of Search Problems

- **Shortest-path (M&C, vacuum, tile puzzle)**
  - want least-cost path to a goal
  - goal depth unknown
  - given operators and their costs

- **Constraint satisfaction (map coloring, $n$-queens)**
  - any goal is fine
  - maximum depth = number of variables
  - given explicit constraints on variables

- **Combinatorial optimization (TSP, max-CSP)**
  - want least-cost goal
  - maximum depth = number of variables
  - every leaf is a solution
A tree representation of alternatives in a small combinatorial problem.
Backtracking

depth-first search
child ordering
lower bounds
branch-and-bound
**Depth-first Search**

DFS \((node)\)  
1. If is-leaf\((node)\)  
2. Visit\((node)\)  
3. else  
4. For \(i\) from 0 to \(num\)-children  
5. DFS\(\text{child}(node, i)\)
Improved Discrepancy Search

ILDS \((node, allowance, remaining)\)

1. If is-leaf\((node)\)
2. Visit\((node)\)
3. else
4. If allowance > 0
5. ILDS(child\((node, 1), allowance - 1, remaining - 1\))
6. If remaining > allowance
7. ILDS(child\((node, 0), allowance, remaining - 1\))

start with ILDS(root, iteration, max-depth)
The second pass of ILDS visits all leaves with one discrepancy in their path from the root.
Break

- asst 4
- projects
- WWICS event tomorrow 6-8pm Kings N343, free food
  
  *Code: Debugging the Gender Gap* and panel
Sol ← some random solution (probably poor quality).
Do limit times
   New ← random neighbor of Sol.
   If New better than Sol, then Sol ← New.
Sol $\leftarrow$ some random solution (probably poor quality).

Do limit times

\begin{align*}
New & \leftarrow \text{random neighbor of } Sol. \\
\text{If } New \text{ better than } Sol, \\
\text{then } Sol & \leftarrow New.
\end{align*}

Elaborations: best neighbor (aka gradient-descent)
- restarts
- simulated annealing
- population (GAs, ‘go with the winners’)

\[ \text{Wheeler Ruml (UNH) Lecture 9, CS 730 – 12 / 17} \]
Continuous Optimization
Types of Optimization Problems

- **Discrete**
  - finite (often small) set of values per choice
  - planning, CSPs, combinatorial optimization
  - constraints can be implicit or explicit

- **Continuous**
  - real values
  - constraints can be linear, quadratic, ..., non-linear
  - objective can be linear, quadratic, ..., non-linear

- **Mixed discrete / continuous**
  - eg, MIPs, MILPs, ...
  - planning for ‘hybrid systems’
real variables, linear constraints, linear objective

- cheapest diet that meets nutrition guidelines
  - \( y_{vitaminA} = 2.3x_{broccoli} + 1.7x_{carrots} \ldots \)
  - \( cost = 4.99x_{broccoli} + 2.67x_{carrots} \ldots \)
  - minimize \( cost \) subject to \( y_{vitaminA} > 500 \ldots \)

- max flow through network with capacity constraints
- earliest finish time subject to job durations

polynomial time (ellipsoid, Karmarkar’s), but simplex method is popular
CPLEX, Gurobi, Ipsove
**convex programming:** constraints and objective are convex
gons are polynomial time

**quadratic programming:** constraints and objective are quadratic
some forms are polynomial time

**0-1 LP:** 0-1 variables, linear constraints, linear objective
NP-complete

**integer linear programming:** integer variables, linear
constraints and objective
NP-complete

**combinatorial optimization:** variables are discrete
Please write down the most pressing question you have about the course material covered so far and put it in the box on your way out.

*Thanks!*