ACM Lab – HS 2015

Prof. Dr. Angelika Steger

October 1, 2015

Exercise: Count the Faces

- Task: output the number of faces of a planar graph
- Eulers formula: $n - m + f = 1 + c$
- Compute the number of components using DFS

Exercise: Count the Faces

- Task: output the number of faces of a planar graph
- Eulers formula: $n - m + f = 1 + c$
- Compute the number of components using DFS

Exercise: Count the Faces

- Task: output the number of faces of a planar graph
- Eulers formula: $n - m + f = 1 + c$
- Compute the number of components using DFS

Exercise: Longest Path

- Task: output the length of the longest path in a graph
- Start BFS at some vertex u and compute vertex v with maximal distance.
- Start BFS at v, the maximal distance to v is the result. Why?
- Yout can also use 2 DFS or a modified DFS...

Exercise: Longest Path

- Task: output the length of the longest path in a graph
- Start BFS at some vertex u and compute vertex v with maximal distance.
- Start BFS at v, the maximal distance to v is the result. Why?
- Yout can also use 2 DFS or a modified DFS...

Exercise: Longest Path

- Task: output the length of the longest path in a graph
- Start BFS at some vertex u and compute vertex v with maximal distance.
- Start BFS at v, the maximal distance to v is the result. Why?
- Yout can also use 2 DFS or a modified DFS...

Exercise: Longest Path

- Task: output the length of the longest path in a graph
- Start BFS at some vertex u and compute vertex v with maximal distance.
- Start BFS at v, the maximal distance to v is the result. Why?
- Yout can also use 2 DFS or a modified DFS...

Exercise: Racetracks

- Task: find the shortest path in an implicit graph
- BFS

Exercise: Racetracks

- Task: find the shortest path in an implicit graph
- BFS

Dynamic Programming

- Formulate problem as recursion
- Trick: compute subproblems only once:
  - Top-Down: add memoization
  - Bottom-Up: build table

Dynamic Programming

- Formulate problem as recursion
- Trick: compute subproblems only once:
  - Top-Down: add memoization
  - Bottom-Up: build table

Dynamic Programming

- Formulate problem as recursion
- Trick: compute subproblems only once:
  - Top-Down: add memoization
  - Bottom-Up: build table

Dynamic Programming

- Formulate problem as recursion
- Trick: compute subproblems only once:
  - Top-Down: add memoization
  - Bottom-Up: build table

Example: Fibonacci Numbers

$f(1) = 1$, $f(2) = 1$, and $f(n) = f(n-1) + f(n-2)$