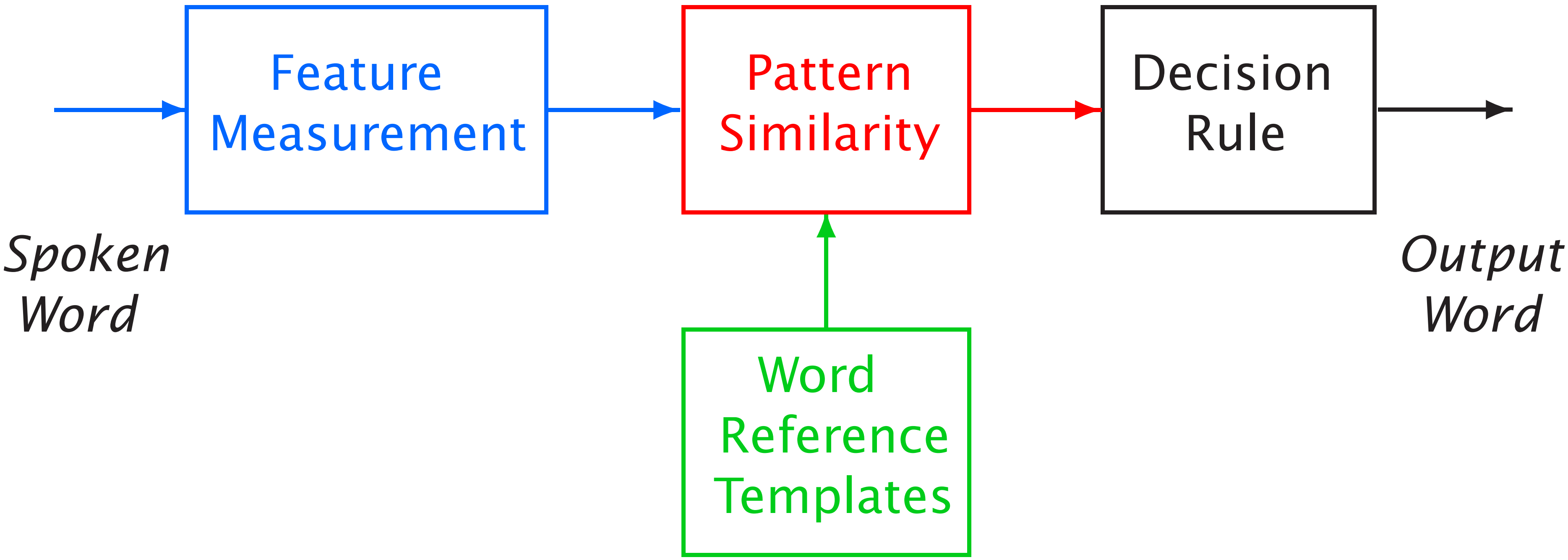
Dynamic Time Warping & Search

- Dynamic time warping
- Search
  - Graph search algorithms
  - Dynamic programming algorithms

Word-Based Template Matching



- Whole word representation:
  - No explicit concept of sub-word units (e.g., phones)
  - No across-word sharing
- Used for both isolated- and connected-word recognition
- Popular in late 1970s to mid 1980s

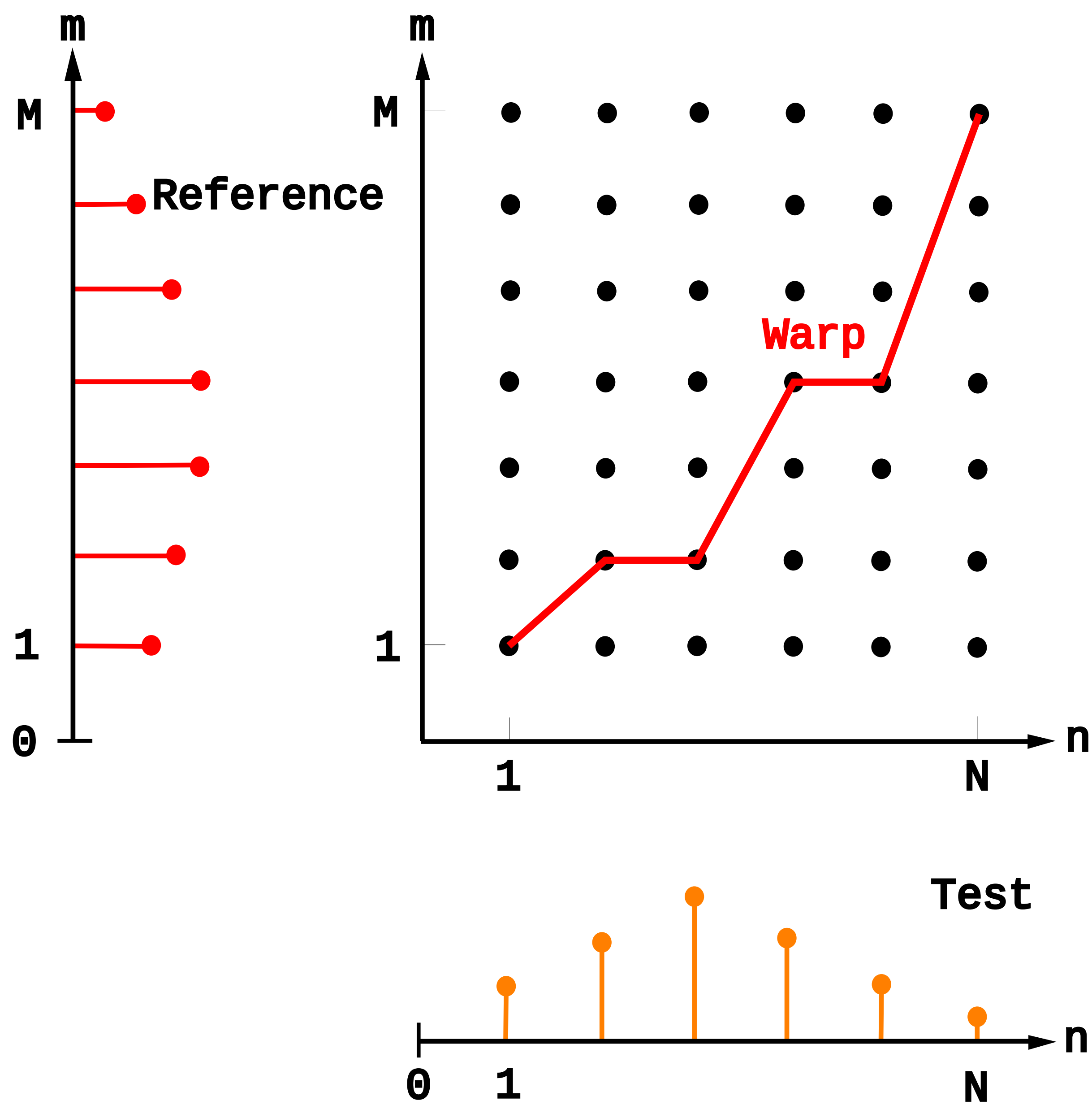
Template Matching Mechanism

- Test pattern, $\boldsymbol{T}$, and reference patterns, $\{\boldsymbol{R}_1, \ldots, \boldsymbol{R}_V\}$, are represented by sequences of feature measurements
- Pattern similarity is determined by aligning test pattern, $\boldsymbol{T}$, with reference pattern, $\boldsymbol{R}_v$, with distortion $\mathcal{D}(\boldsymbol{T}, \boldsymbol{R}_v)$
- Decision rule chooses reference pattern, $\boldsymbol{R}^*$, with smallest alignment distortion $\mathcal{D}(\boldsymbol{T}, \boldsymbol{R}^*)$

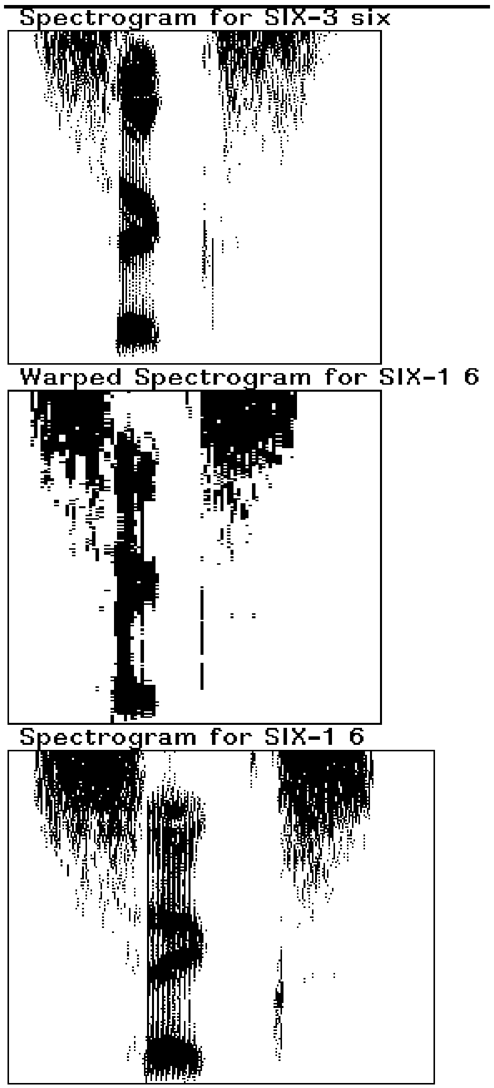
$$\boldsymbol{R}^* = \arg\min_v \mathcal{D}(\boldsymbol{T}, \boldsymbol{R}_v)$$

- Dynamic time warping (DTW) is used to compute the best possible alignment warp, ϕ_v, between $\boldsymbol{T}$ and $\boldsymbol{R}_v$, and the associated distortion $\mathcal{D}(\boldsymbol{T}, \boldsymbol{R}_v)$

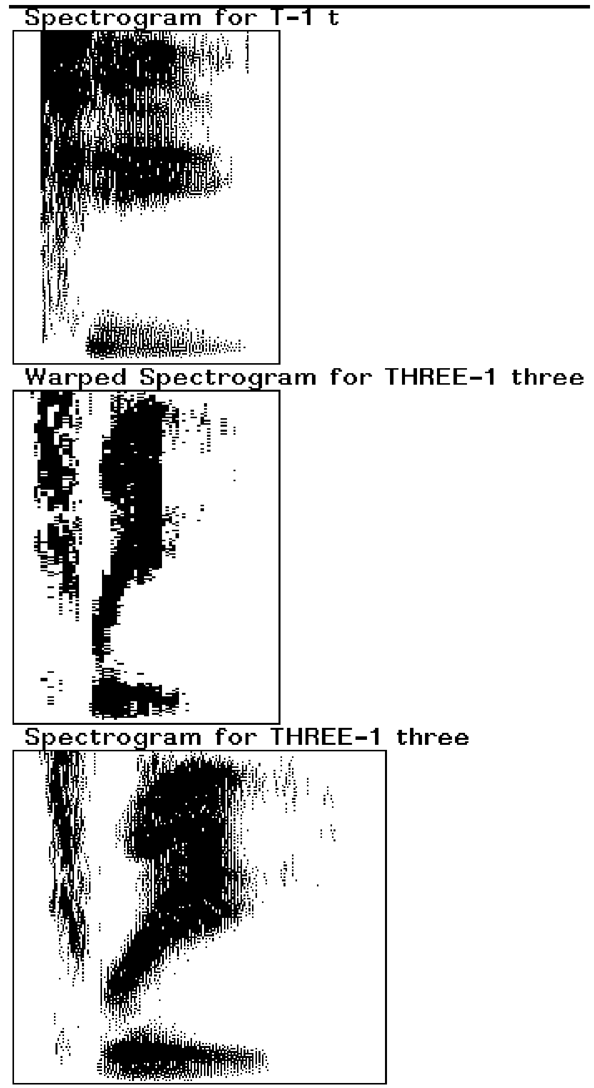
Alignment Example



Digit Alignment Examples



Match



Mismatch

Dynamic Time Warping (DTW)

- Objective: an optimal alignment between variable length sequences $\boldsymbol{T} = \{\boldsymbol{t}_1, \ldots, \boldsymbol{t}_N\}$ and $\boldsymbol{R} = \{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_M\}$
- The overall distortion $\mathcal{D}(\boldsymbol{T}, \boldsymbol{R})$ is based on a sum of local distances between elements $d(\boldsymbol{t}_i, \boldsymbol{r}_j)$
- A particular alignment warp, ϕ, aligns $\boldsymbol{T}$ and $\boldsymbol{R}$ via a point-to-point mapping, $\phi = (\phi_t, \phi_r)$, of length K_ϕ

$$\boldsymbol{t}_{\phi_t(k)} \Longleftrightarrow \boldsymbol{r}_{\phi_r(k)} \quad 1 \le k \le K_\phi$$

- The optimal alignment minimizes overall distortion

$$\mathcal{D}(\boldsymbol{T}, \boldsymbol{R}) = \min_{\phi} \mathcal{D}_\phi(\boldsymbol{T}, \boldsymbol{R})$$

$$\mathcal{D}_\phi(\boldsymbol{T}, \boldsymbol{R}) = \frac{1}{M_\phi} \sum_{k=1}^{K_\phi} d(\boldsymbol{t}_{\phi_t(k)}, \boldsymbol{r}_{\phi_r(k)}) m_k$$

DTW Issues

- Endpoint constraints:

$$\phi_t(1) = \phi_r(1) = 1 \quad \phi_t(K) = N \quad \phi_r(K) = M$$

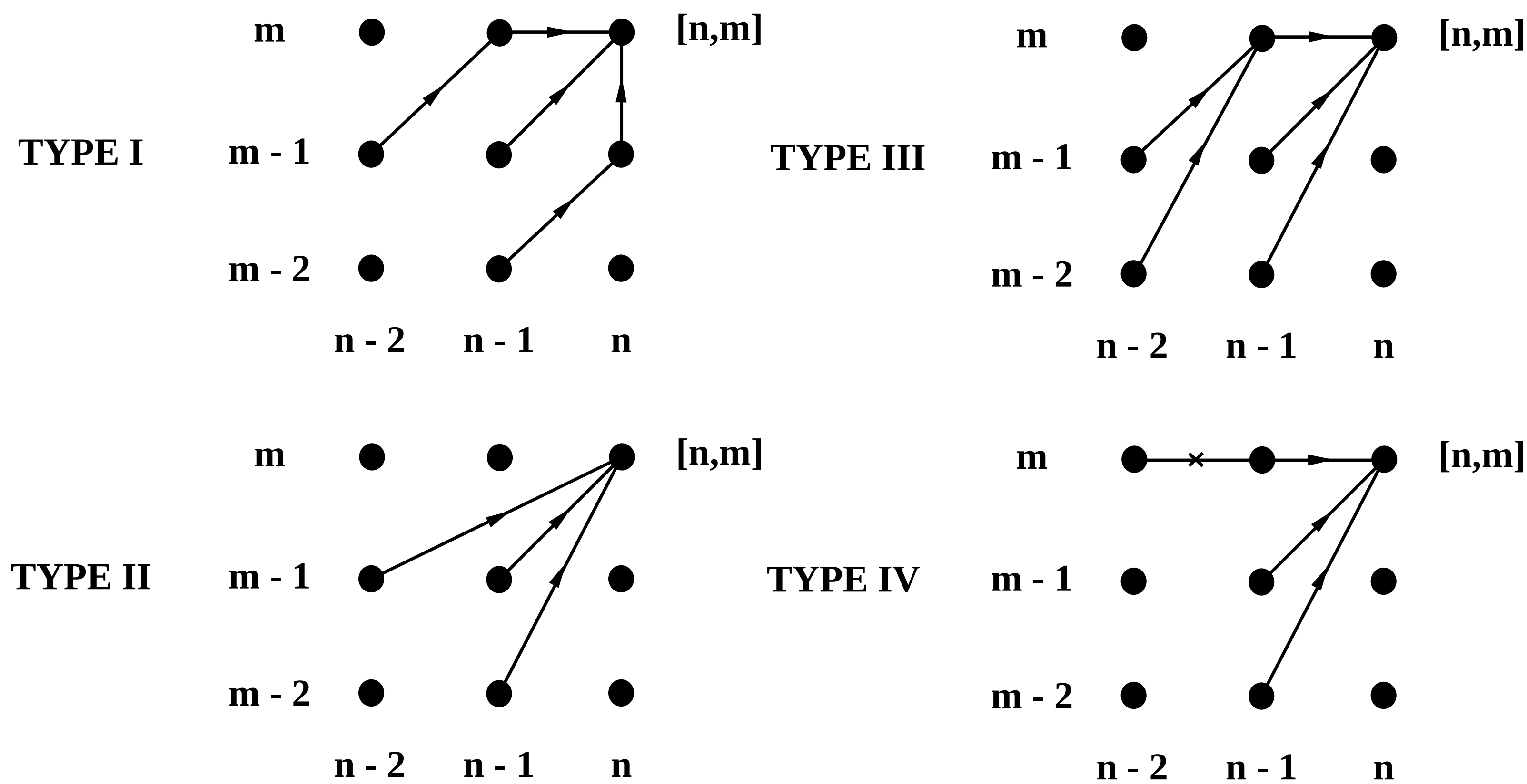
- Monotonicity:

$$\phi_t(k+1) \geq \phi_t(k) \quad \phi_r(k+1) \geq \phi_r(k)$$

- Path weights, m_k, can influence shape of optimal path
- Path normalization factor, M_ϕ, allows comparison between different warps (e.g., with different lengths)

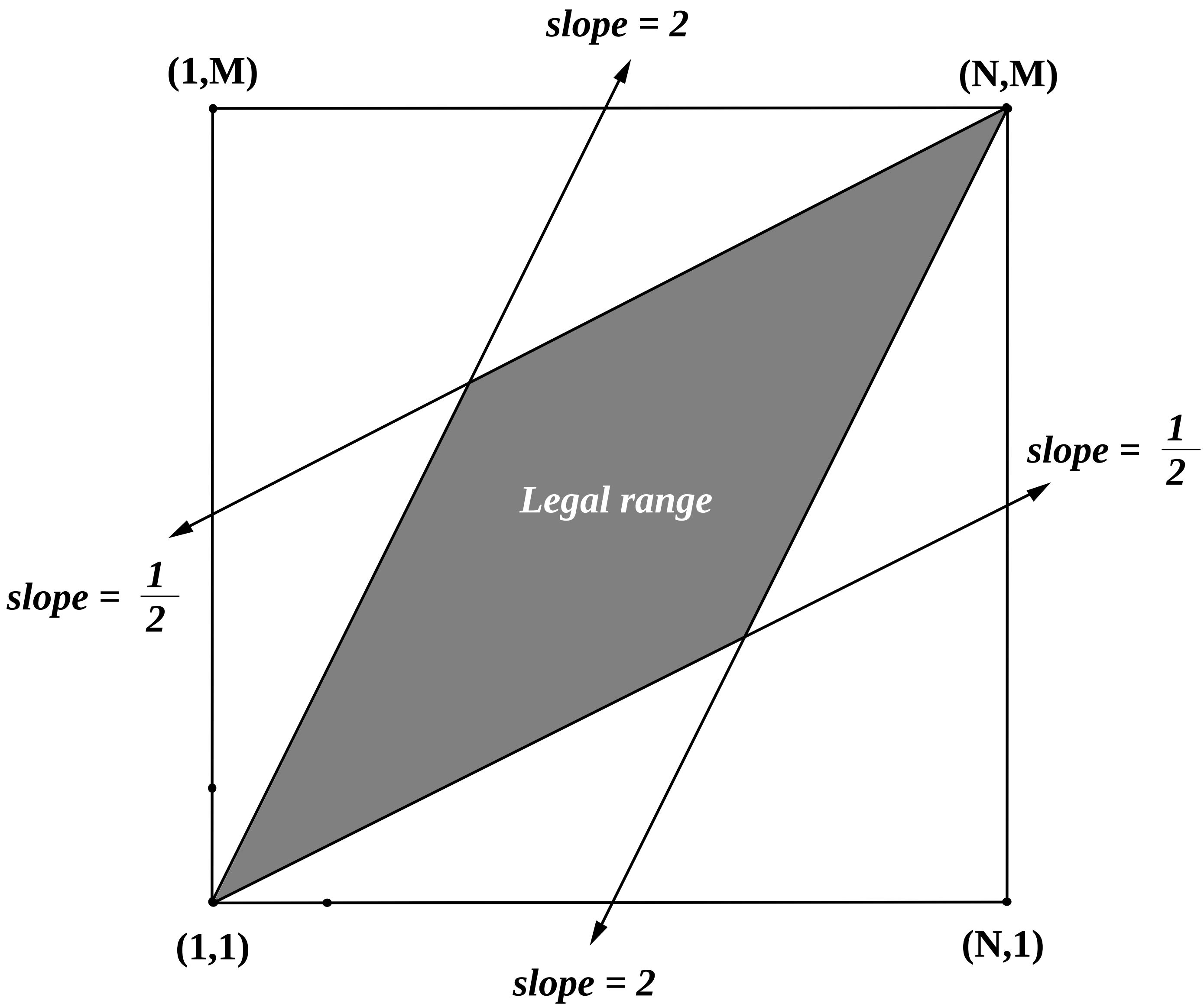
$$M_\phi = \sum_{k=1}^{K_\phi} m_k$$

DTW Issues: Local Continuity



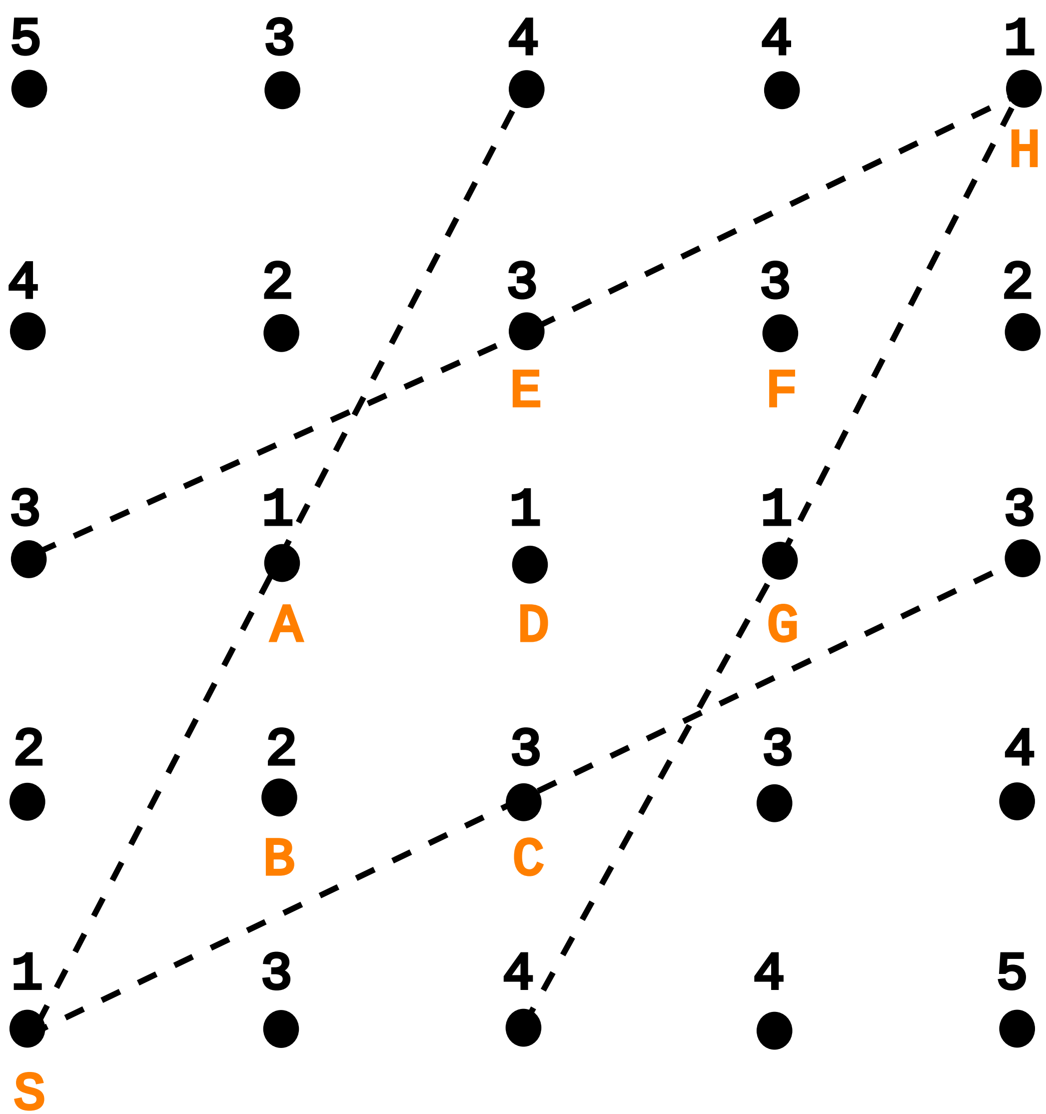
Local constraints determine alignment flexibility

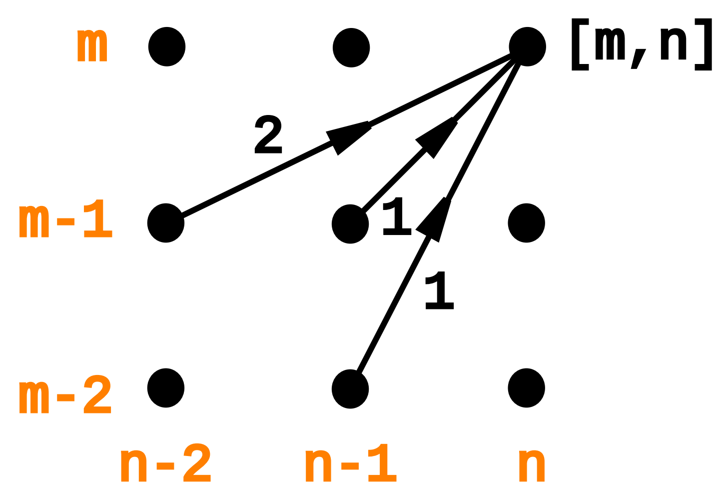
DTW Issues: Global Constraints



Local constraints exclude portions of search space

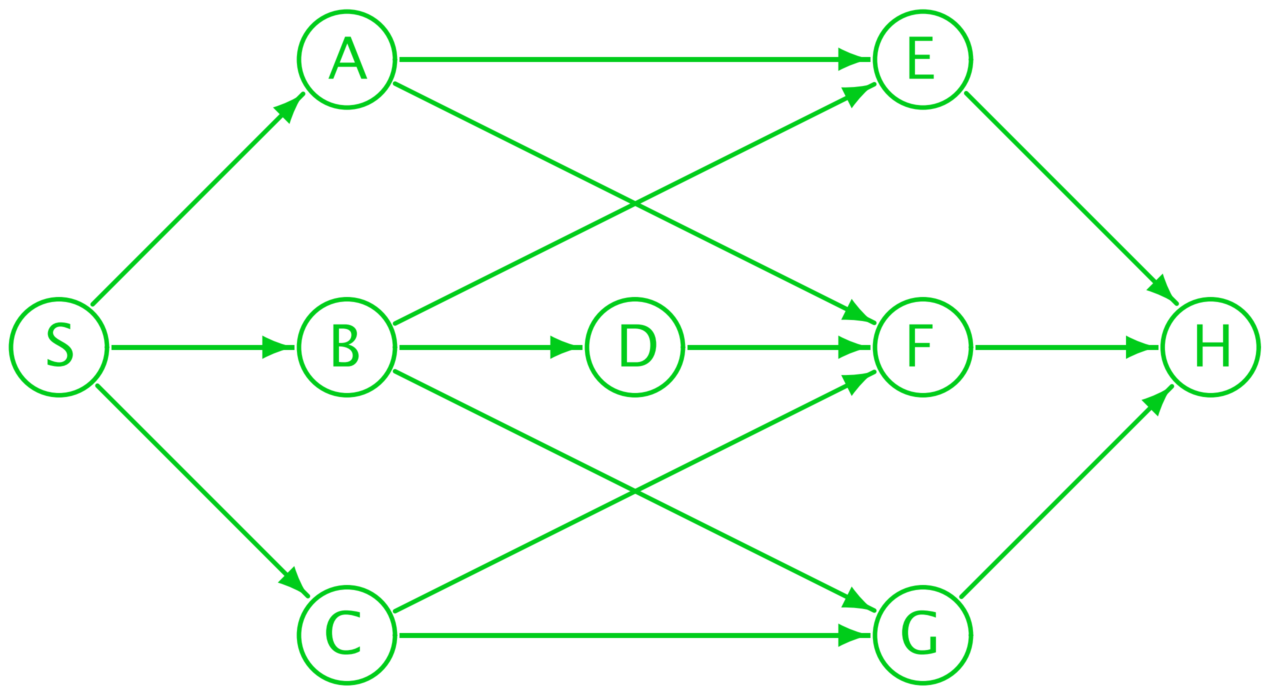
Computing DTW Alignment





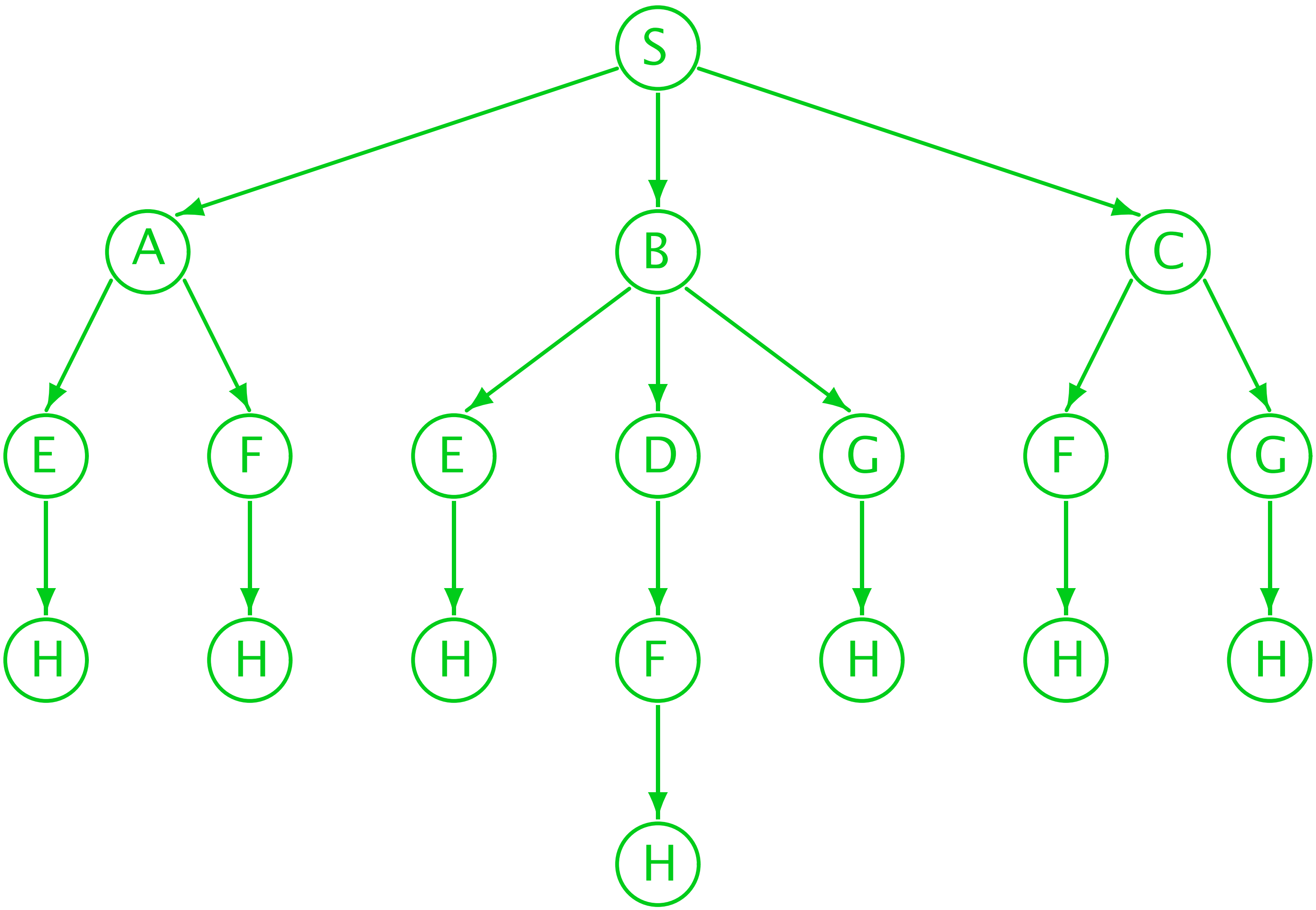
Graph Representations of Search Space

- Search spaces can be represented as directed graphs



- Paths through a graph can be represented with a tree

Search Space Tree



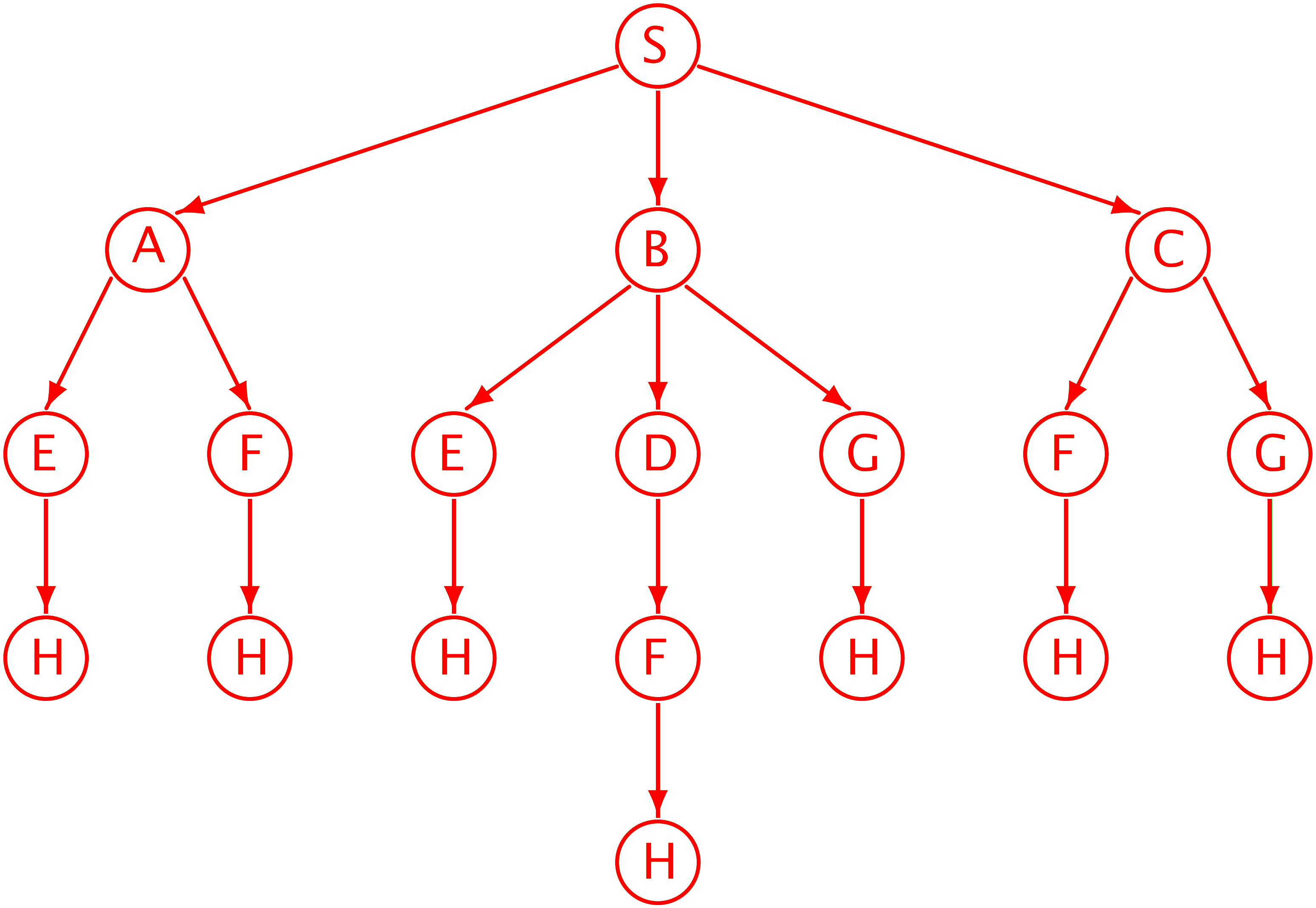
Graph Search Algorithms

- Iterative methods using a queue to store partial paths
  - On each iteration the top partial path is removed from the queue and is extended one level
  - New extensions are put back into the queue
  - Search is complete when goal is reached
- Depth of queue is potentially unbounded
- Weighted graphs can be searched to find the best path
- Admissible algorithms guarantee finding the best path
- Many speech-based search problems can be configured to proceed time-synchronously

Depth First Search

- Searches space by pursuing one path at a time
- Path extensions are put on top of queue
- Queue is not reordered or pruned
- Not well suited for spaces with long dead-end paths
- Not generally used to find the best path

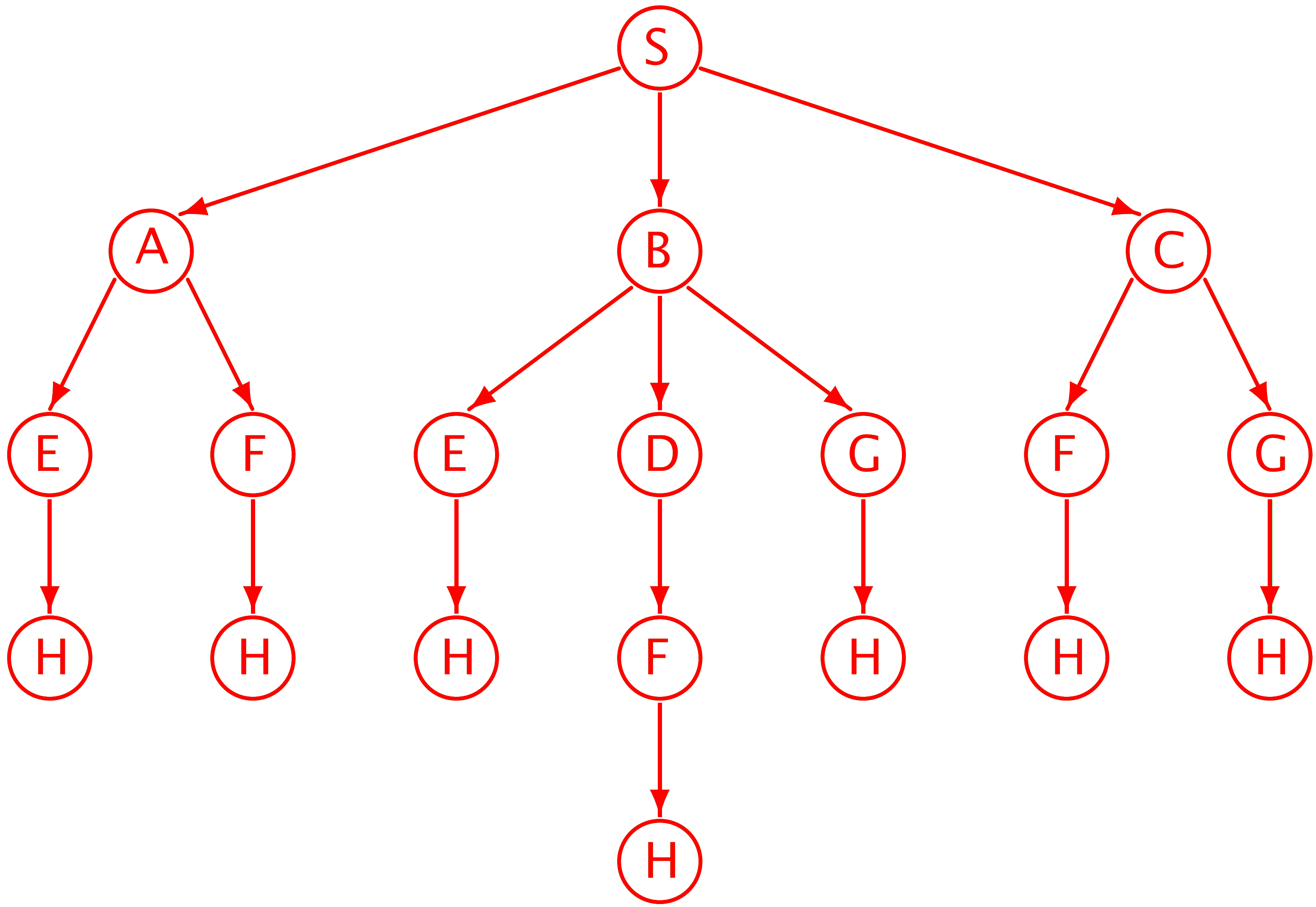
Depth First Search Example



Breadth First Search

- Searches space by pursuing all paths in parallel
- Path extensions are put on bottom of queue
- Queue is not reordered or pruned
- Queue can grow rapidly in spaces with many paths
- Not generally used to find the best path
- Can be made much more effective with pruning

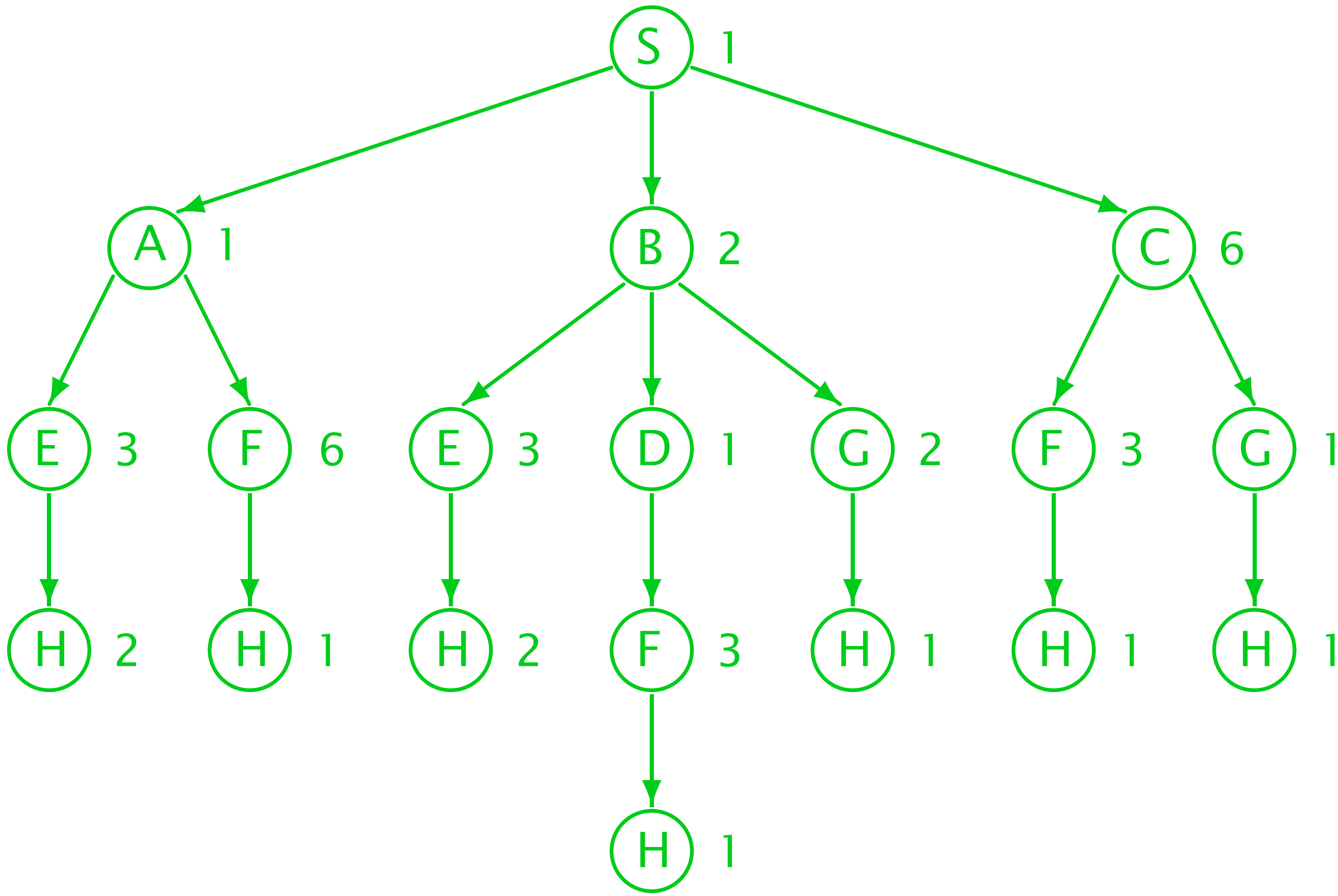
Breadth First Search Example



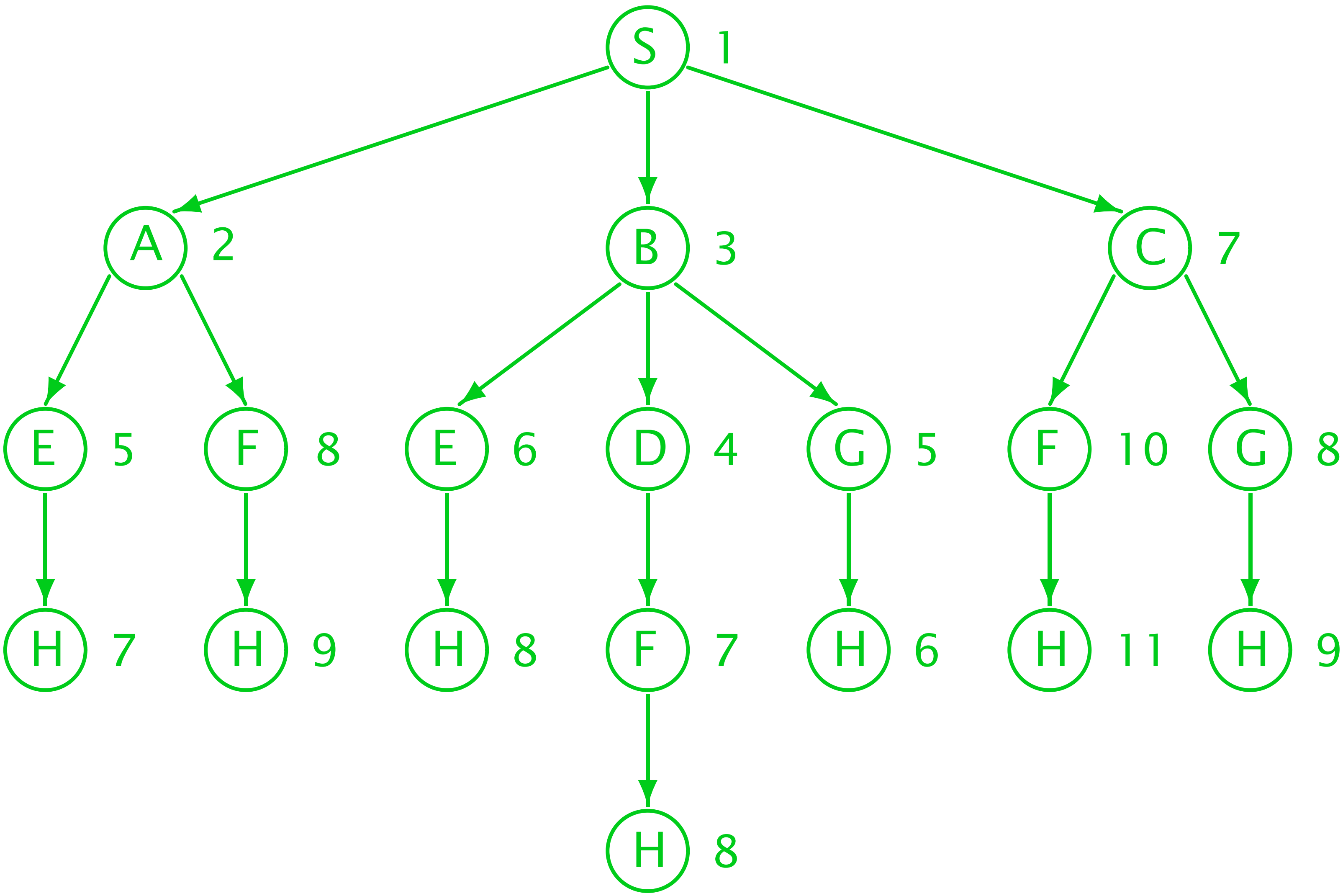
Best First Search

- Used to search a weighted graph
- Uses greedy or step-wise optimal criterion, whereby each iteration expands the current best path
- On each iteration, the queue is resorted according to the cumulative score of each partial path
- If path scores exhibit monotonic behavior, (e.g., $d(\boldsymbol{t}_i, \boldsymbol{r}_j) \geq 0$), search can terminate when a complete path has a better score than all active partial paths

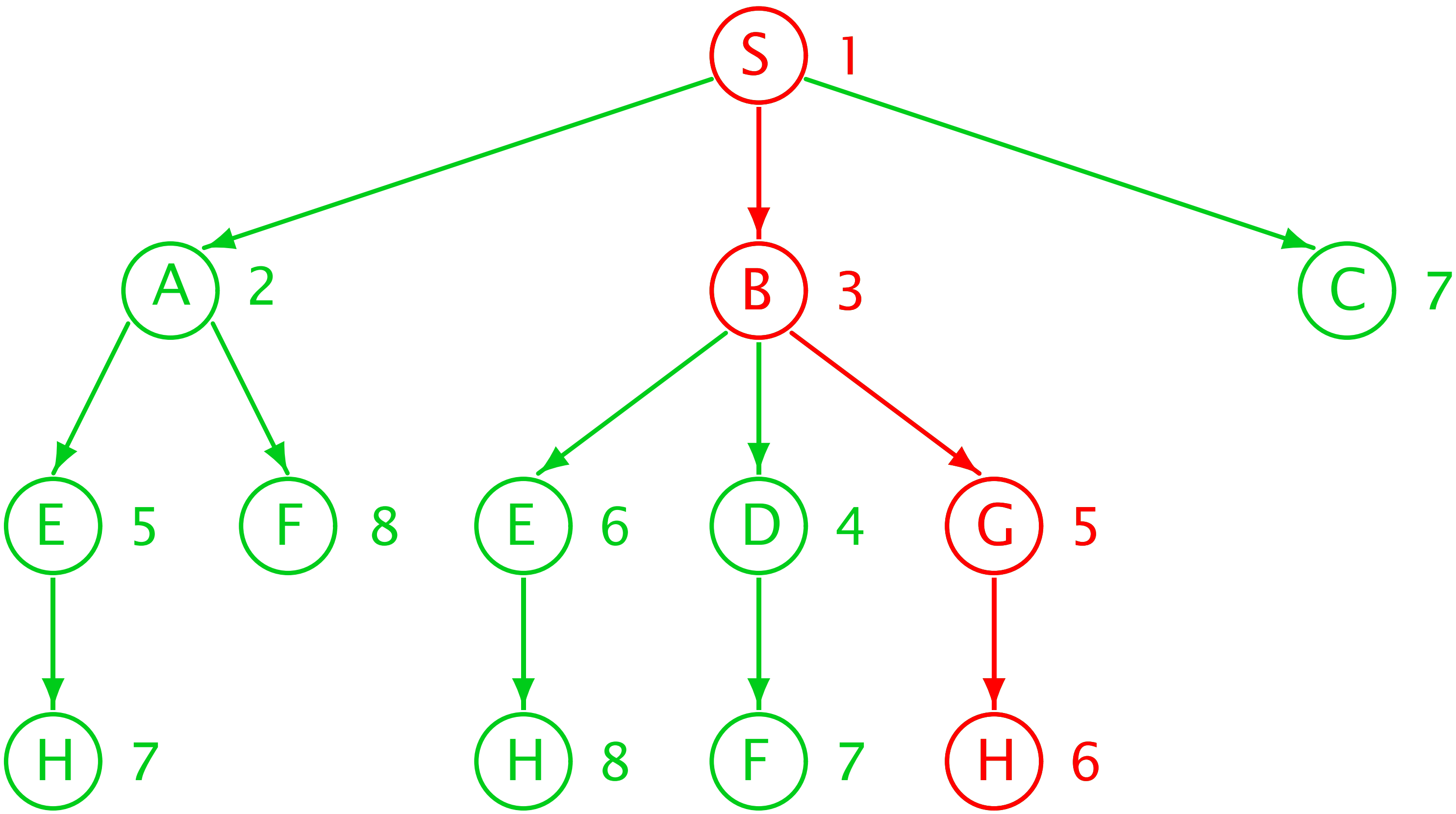
Tree Representation (with node scores)



Tree Representation (with cumulative scores)



Best First Search Example



Pruning Partial Paths

- Both greedy and dynamic programming algorithms can take advantage of optimal substructure:
  - Let $\phi(i,j)$ be the best path between nodes i and j
  - If k is a node in $\phi(i,j)$:

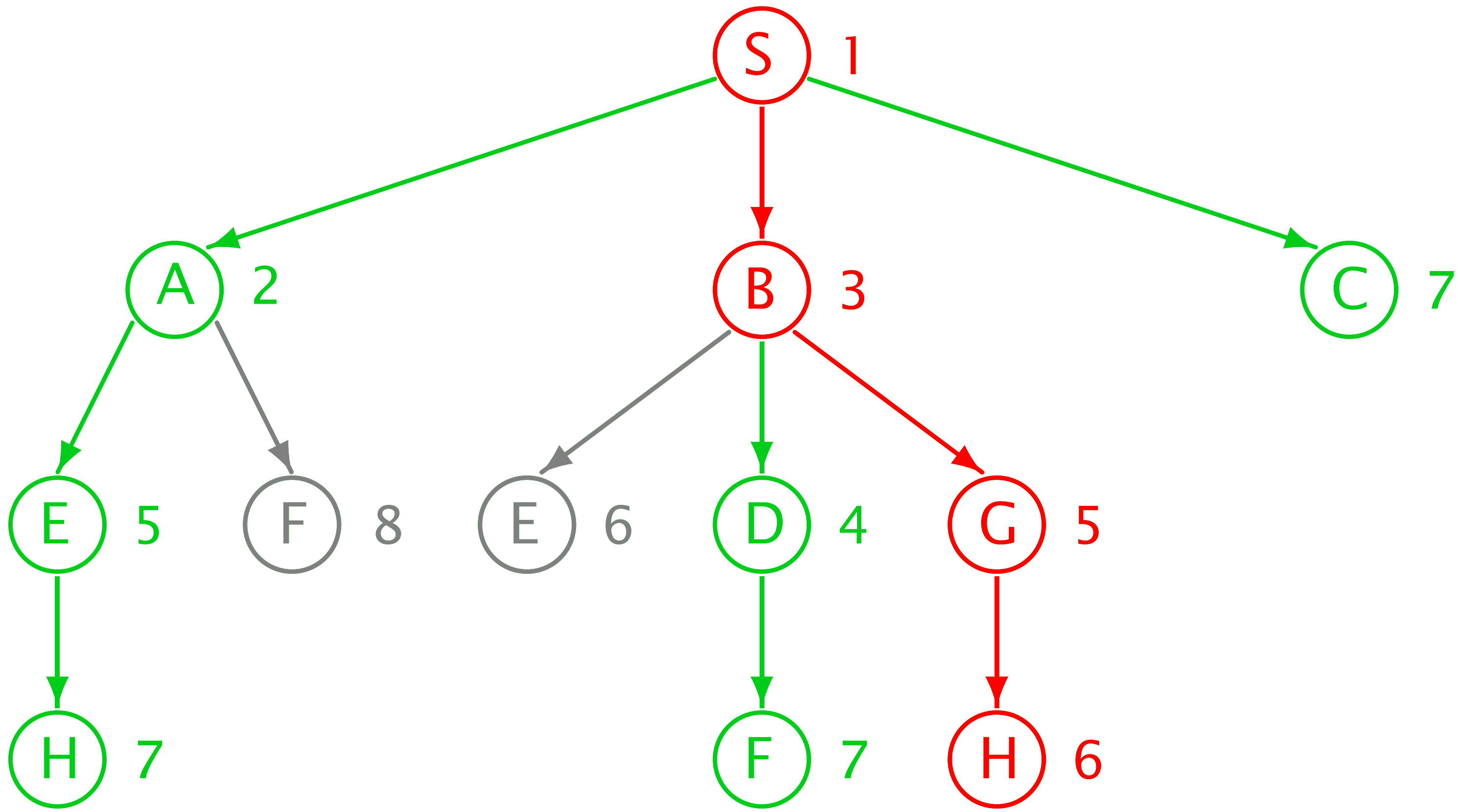
$$\phi(i,j) = \{\phi(i,k), \phi(k,j)\}$$

  - Let $\varphi(i,j)$ be the cost of $\phi(i,j)$

$$\varphi(i,j) = \min_k(\varphi(i,k) + \varphi(k,j))$$

- Solutions to subproblems need only be computed once
- Sub-optimal partial paths can be discarded while maintaining admissibility of search

Best First Search with Pruning



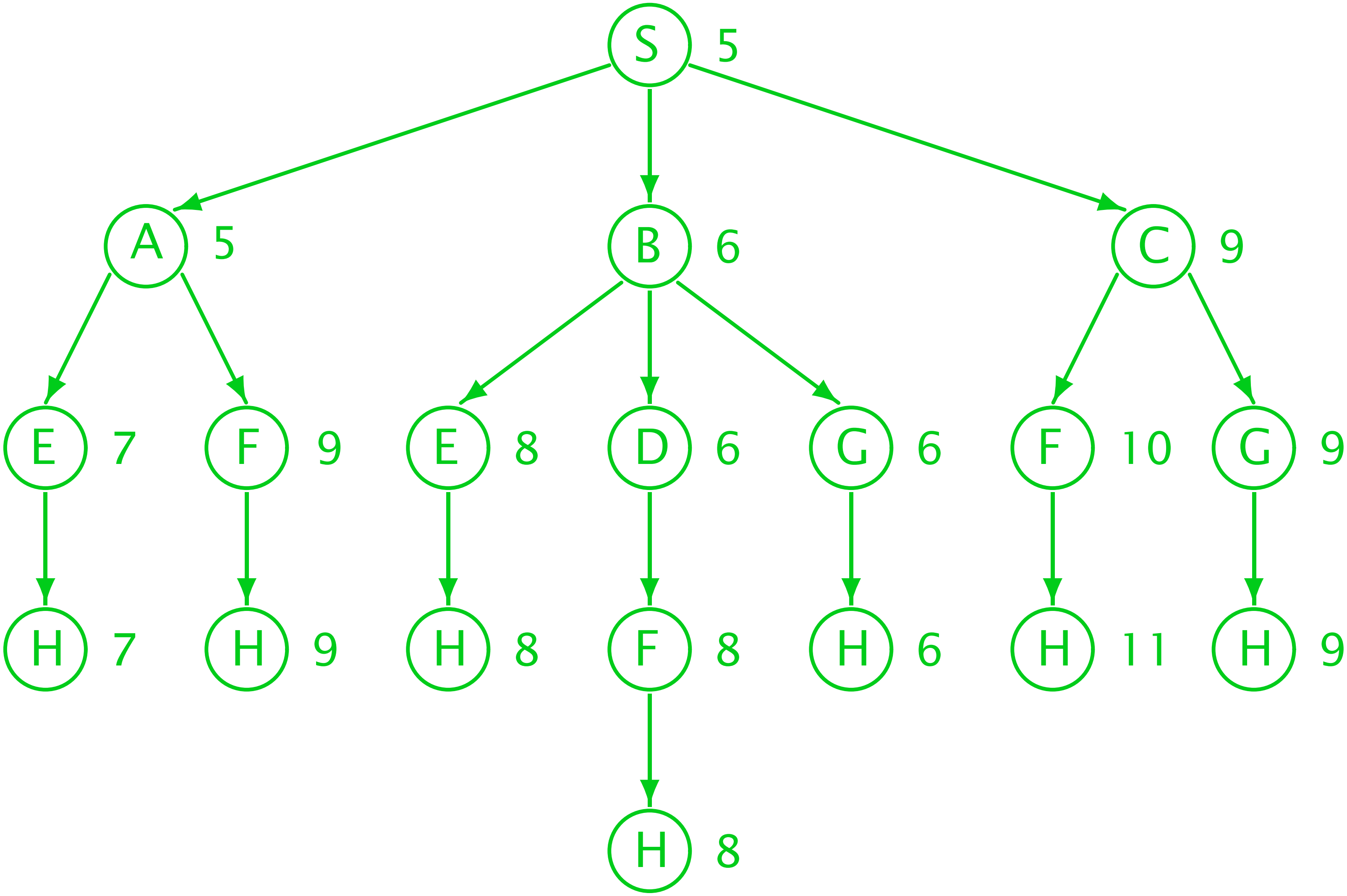
Estimating Future Scores

- Partial path scores, $\varphi(1, i)$, can be augmented with future estimates, $\hat{\varphi}(i)$, of the remaining cost

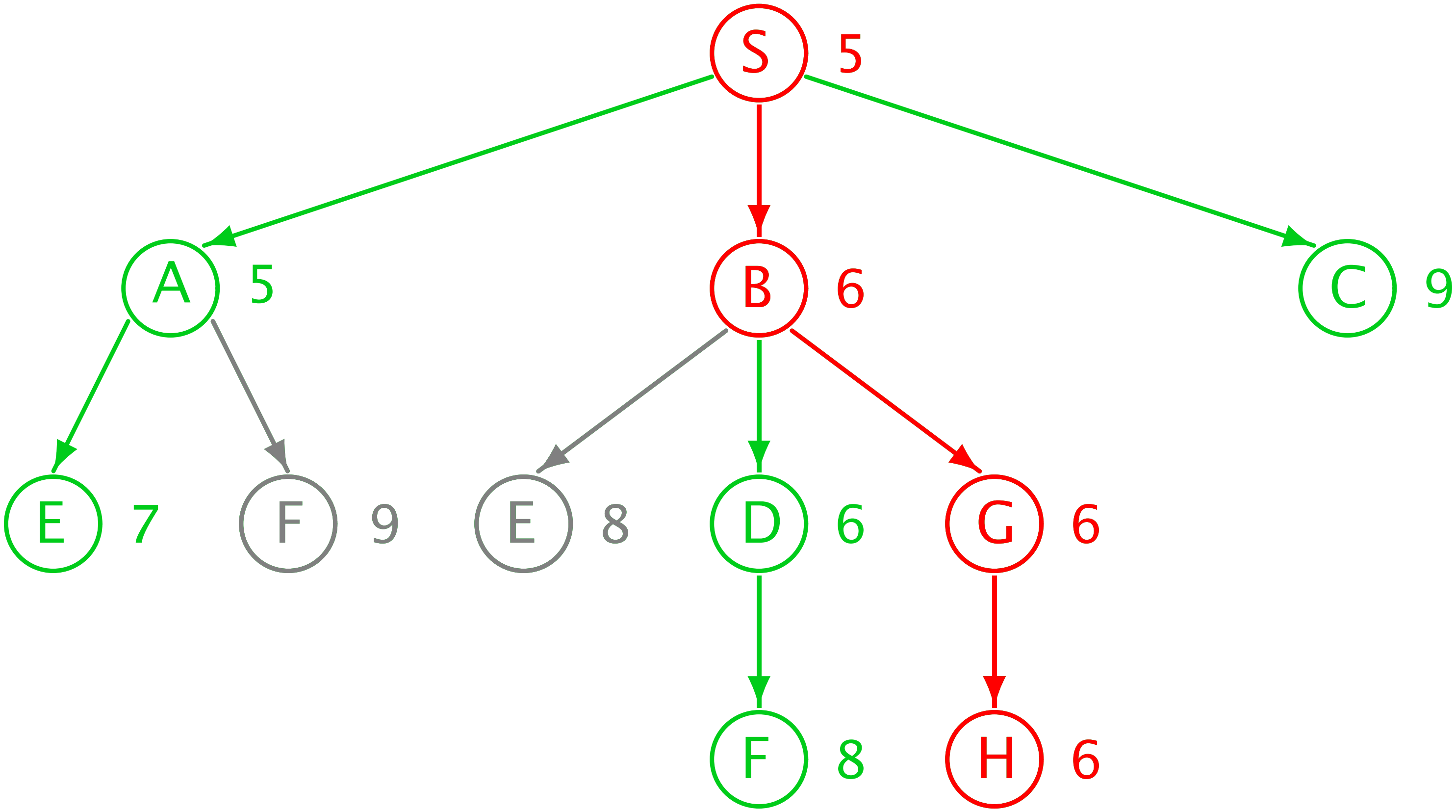
$$\varphi_\phi = \varphi(1, i) + \hat{\varphi}(i)$$

- If $\hat{\varphi}(i)$ is an underestimate of the remaining cost, additional paths can be pruned while maintaining admissibility of search
- A^* search uses
  - Best-first search strategy
  - Pruning
  - Future estimates

Tree Representation (with future estimates)



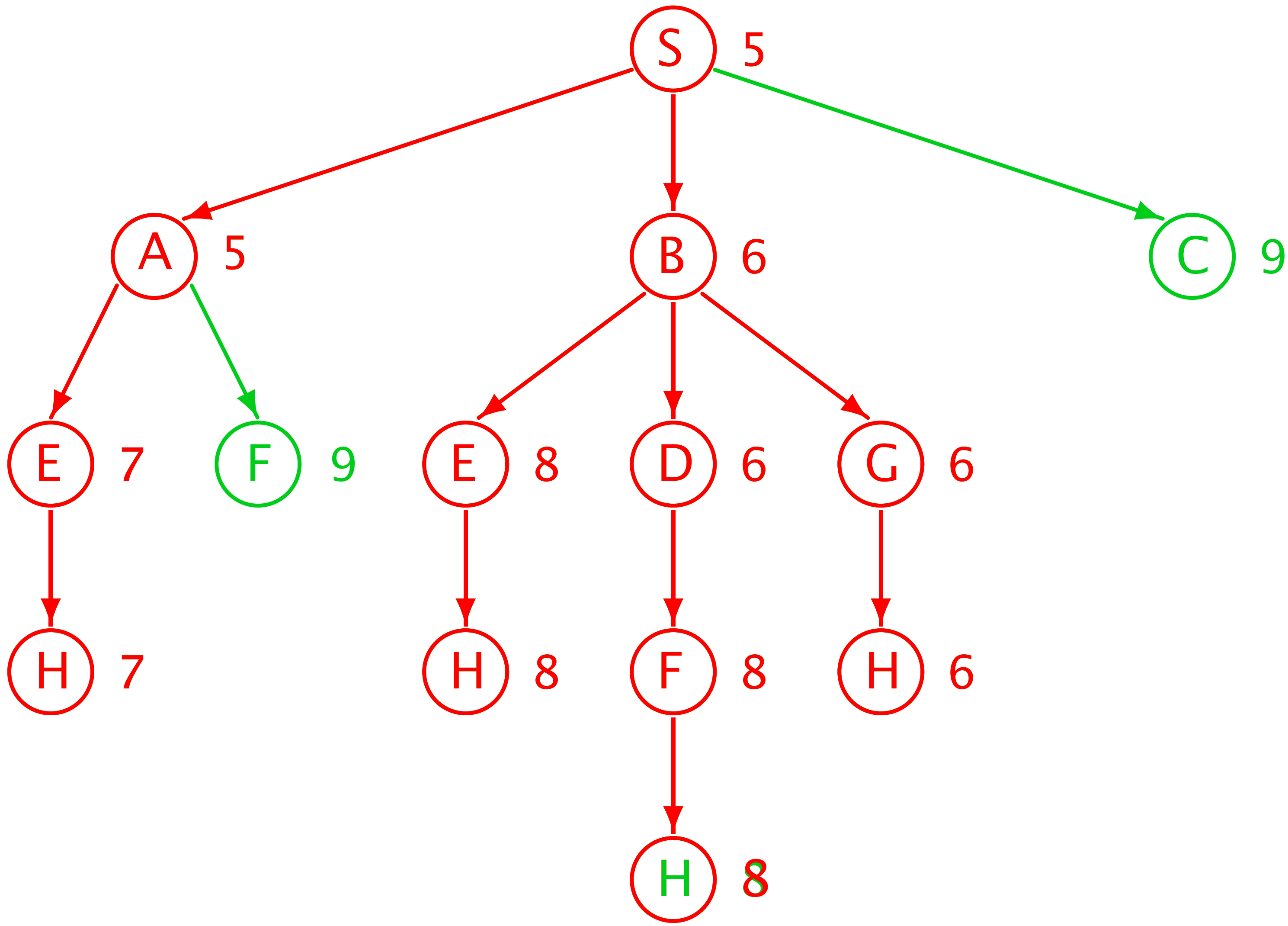
A^* Search Example



N-Best Search

- Used to compute top N paths
  - Can be re-scored by more sophisticated techniques
  - Typically used at the sentence level
- Can use modified A^* search to rank paths
  - No pruning of partial paths
  - Completed paths are removed from queue
  - Can use a threshold to prune paths, and still identify admissibility violations
  - Can also be used to produce a graph
- Alternative methods can be used to compute N-best outputs (e.g., asynchronous DP)

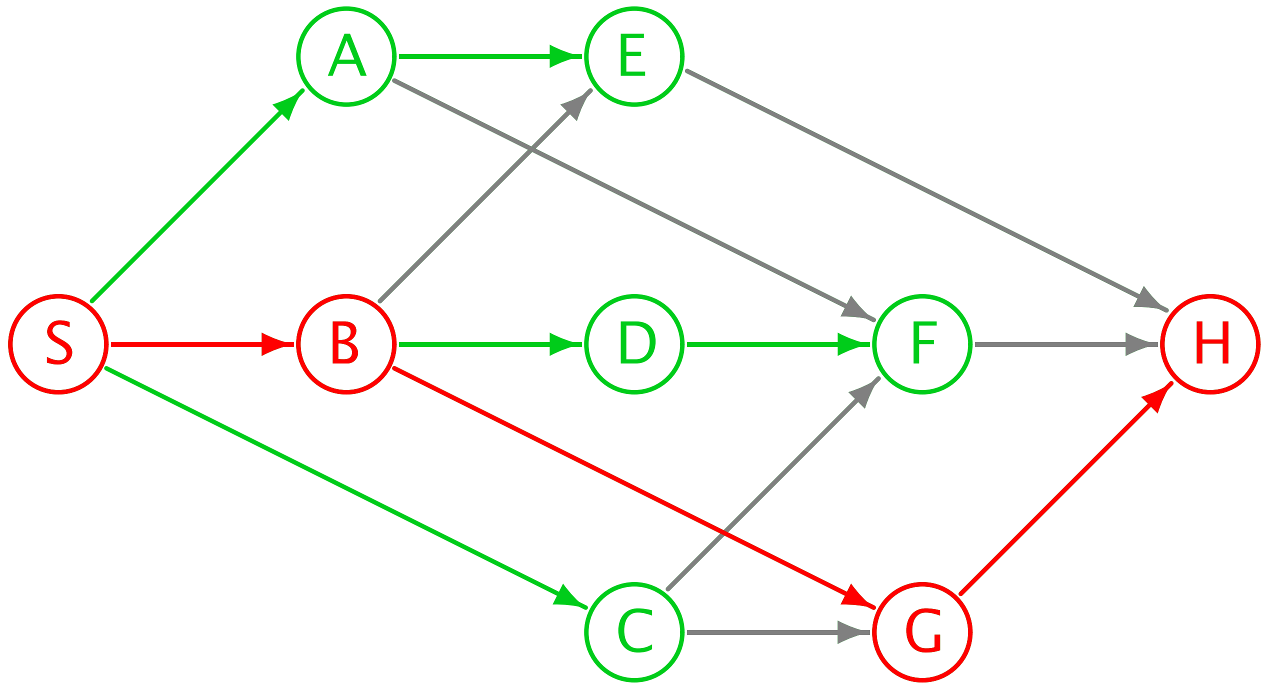
N-Best Search Example



Dynamic Programming (DP)

- DP algorithms do not employ a greedy strategy
- DP algorithms typically take advantage of optimal substructure and overlapping subproblems by arranging search to solve each subproblem only once
- Can be implemented efficiently:
  - Node j retains only best path cost of all $\varphi(i, j)$
  - Previous best node id needed to recover best path
- Can be time-synchronous or asynchronous
- DTW and Viterbi are time-synchronous searches and look like breadth-first with pruning

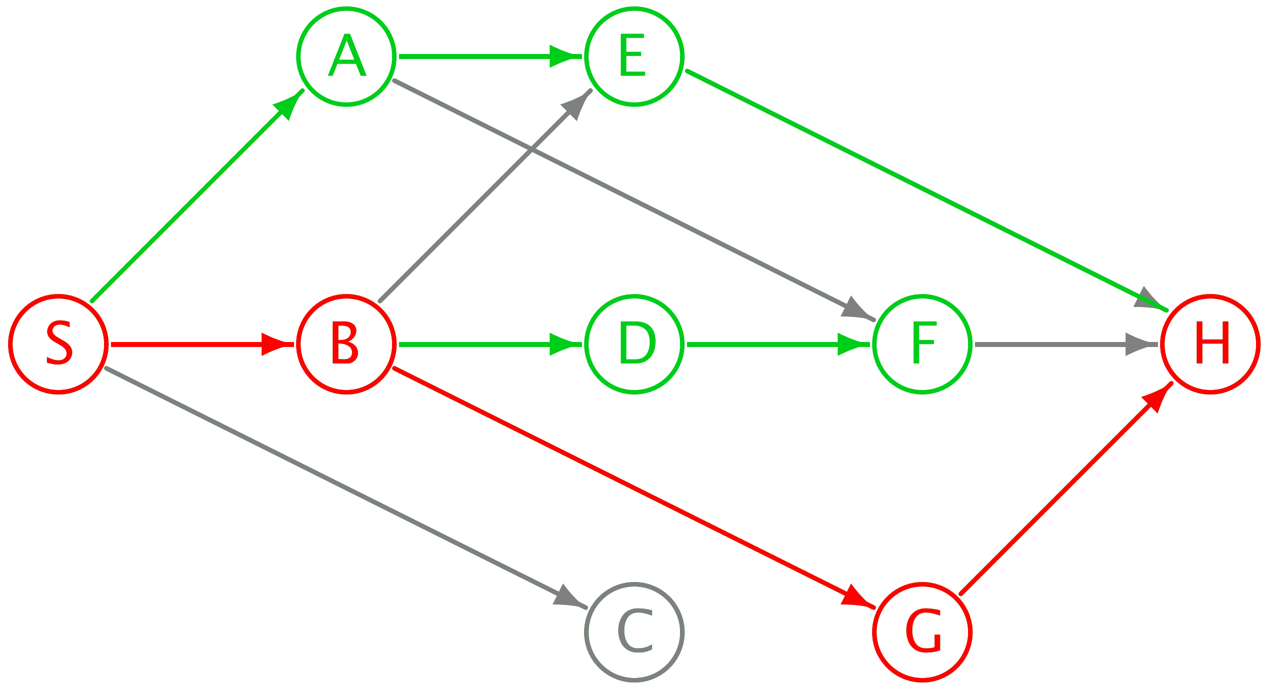
Time-Synchronous DP Example



Inadmissible Search Variations

- Can use a beam width to prune current hypotheses
  - Beam width can be static or dynamic based on relative score
- Can use an approximation to a lower bound on A^* lookahead for N-best computation
- Search is inadmissible, but may be useful in practice

Beam Search Example



References

- Cormen, Leiserson, Rivest, *Introduction to Algorithms*, 2^{nd} Edition, MIT Press, 2001.
- Huang, Acero, and Hon, *Spoken Language Processing*, Prentice-Hall, 2001.
- Jelinek, *Statistical Methods for Speech Recognition.* MIT Press, 1997.
- Winston, *Artificial Intelligence,* 3^{rd} Edition, Addison-Wesley, 1993.