Algorithm

Yangjun Ahn
yangjunahn@sungshin.ac.kr
https://sites.google.com/sungshin.ac.kr/mhail

성신여자대학교

Shortest Path

- **$\delta(s, v)$ for all v**
- **$$\delta(s, v) = \min_{(u, v) \in E} \big(\underbrace{\delta(s, v)}_{\text{recursive call}} + w(u, v) \big)$$**
- **Is this good algorithm?**
- **Subproblem dependencies should be acyclic.**
- **Time?**
  - $\Theta(V^3)$
  - but actually? $\Theta(VE)$ ▶ Bellman-Ford

AI융합학부 안양준 yangjunahn@sungshin.ac.kr

Bellman-Ford

- Generic shortest path algorithm.
- Analysis
- Correctness

Bellman-Ford Algorithm

```
// r : starting vertex
Bellman-Ford(G, r) {
    for(u ∈ V)
        d[u] = ∞;
    d[r] = 0;
    // Calculate shortest path : T(V, E) = O(|V| * |E|)
    for(int i=1; i <= |V|-1; i++)
        for((u, v) ∈ E)
            if(d[u] + w(u, v) < d[v]) {
                d[v] = d[u] + w(u, v);
                prev[v] = u; }
    // Check the negative weighted cycle : T(V, E) = O(|E|)
    for((u, v) ∈ E)
        if(d[u] + w(u, v) < d[v])
            print "A negative weighted cycle exists."; }
```

Verification

- **Pigeonhole Principle**
  - Birthday problem
- **Theorem**

After the for loop ($i = m$), which is the m th iteration, ends, the shortest path that can be obtained using up to m edges is calculated.

▶ **Induction? Do this with me.**

Easy Dynamic Programming

- Three main items: guess, recursion, and memorization
- Remember

  Time = (# subprobs) x (time / subprob)

- Five generic (easy!) steps:
  1. Define subproblems
  2. Guess your possible choices
  3. Ideate subproblems
  4. Apply recursion and memorization
  5. Solve the original problem

AI융합학부 안양준 yangjunahn@sungshin.ac.kr

Fill in the Blank

	Fibonacci	Shortest Path	Knapsack
1 Define subprob.			
2 Guess			
3 Ideate			
4 Rec + memo			
5 Solve orig.			

Time Complexity

- Pseudopolynomial

  = (Polynomial in the problem size)

  x (the numbers in the input)