More Dynamic Programming

Common Subproblems

- Opt(i) - Opt solution using $x_1,..,x_i$. (eg LIS, longest path).
- Opt(i,j) - Opt solution using $x_i,...,x_j$. (eg RNA)
- Opt(i,j) - Opt solution using $x_1,...,x_i$ and $y_1,...,y_j$. (eg Edit distance)
- Opt(r) - Opt solution using subtree rooted at r. (eg Vertex cover on trees).

Longest increasing subsequence

Given: sequence of numbers

Goal: find longest increasing subsequence

41 , 22 , 9 , 15 , 23 , 39 , 21 , 56 , 24 , 34 , 59 , 23 , 60 , 39 , 87 , 23 , 90

Longest increasing subsequence

Given: sequence of numbers

Goal: find longest increasing subsequence

41 , 22 , **9** , **15** , **23** , 39 , 21 , 56 , **24** , **34** , **59** , 23 , **60** , 39 , **87** , 23 , **90**

longest increasing subsequence: length 9

Longest increasing subsequence

Given: sequence of numbers $x_1,..,x_n$

Goal: find longest increasing subsequence

41 , 22 , 9 , 15 , 23 , 39 , 21 , 56 , 24 , 34 , 59 , 23 , 60 , 39 , 87 , 23 , 90

Subproblems: l(j) - length of longest increasing subseq. ending at j.

Longest increasing subsequence

Given: sequence of numbers $x_1,..,x_n$

Goal: find longest increasing subsequence

41 , 22 , 9 , 15 , 23 , 39 , 21 , 56 , 24 , 34 , 59 , 23 , 60 , 39 , 87 , 23 , 90

Subproblems: l(j) - length of longest increasing subseq. ending at j.

Observation: if longest inc. sub. ending at j is $x_{i1},x_{i2},...,x_i,x_j$ then $l(j) = l(i)+1$

Longest increasing subsequence

Given: sequence of numbers $x_1,..,x_n$

Goal: find longest increasing subsequence

41 , 22 , 9 , 15 , 23 , 39 , 21 , 56 , 24 , 34 , 59 , 23 , 60 , 39 , 87 , 23 , 90

Subproblems: l(j) - length of longest increasing subseq. ending at j.

Observation: if longest inc. sub. ending at j is $x_{i1},x_{i2},...,x_i,x_j$ then $l(j) = l(i)+1$

Claim: $l(j) = \begin{cases} 1 & \text{if } x_i \geq x_j \text{, for all } i<j \\ 1+\max\limits_{i:\ i<j,\ x_i<x_j} l(i) & \text{else} \end{cases}$

Longest increasing subsequence

Subproblems: $l(j)$ - length of longest increasing subseq. ending at j.

Claim:
$$l(j) = \begin{cases} 1 & \text{if } x_i \geq x_j \text{, for all } i<j \\ 1+\max\limits_{i:\, i<j,\, x_i<x_j} l(i) & \text{else} \end{cases}$$

Algorithm:

for $j=1,\ldots,n$
  if $x_i \geq x_j$, for all $i<j$, set $l(j) = 1$
  else, set $l(j) = 1+\max\limits_{i:\, i<j,\, x_i<x_j} l(i)$

output $\max\limits_{j} l(j)$

Running time

$O(n^2)$

All pairs shortest path in directed graph with no negative cycles.

Given: directed graph, (possibly negative) edge weights

Goal: find shortest path between every two vertices

Bellman-Ford algorithm can do this in time $O(n^2m)$

All pairs shortest path in directed graph with weighted edges

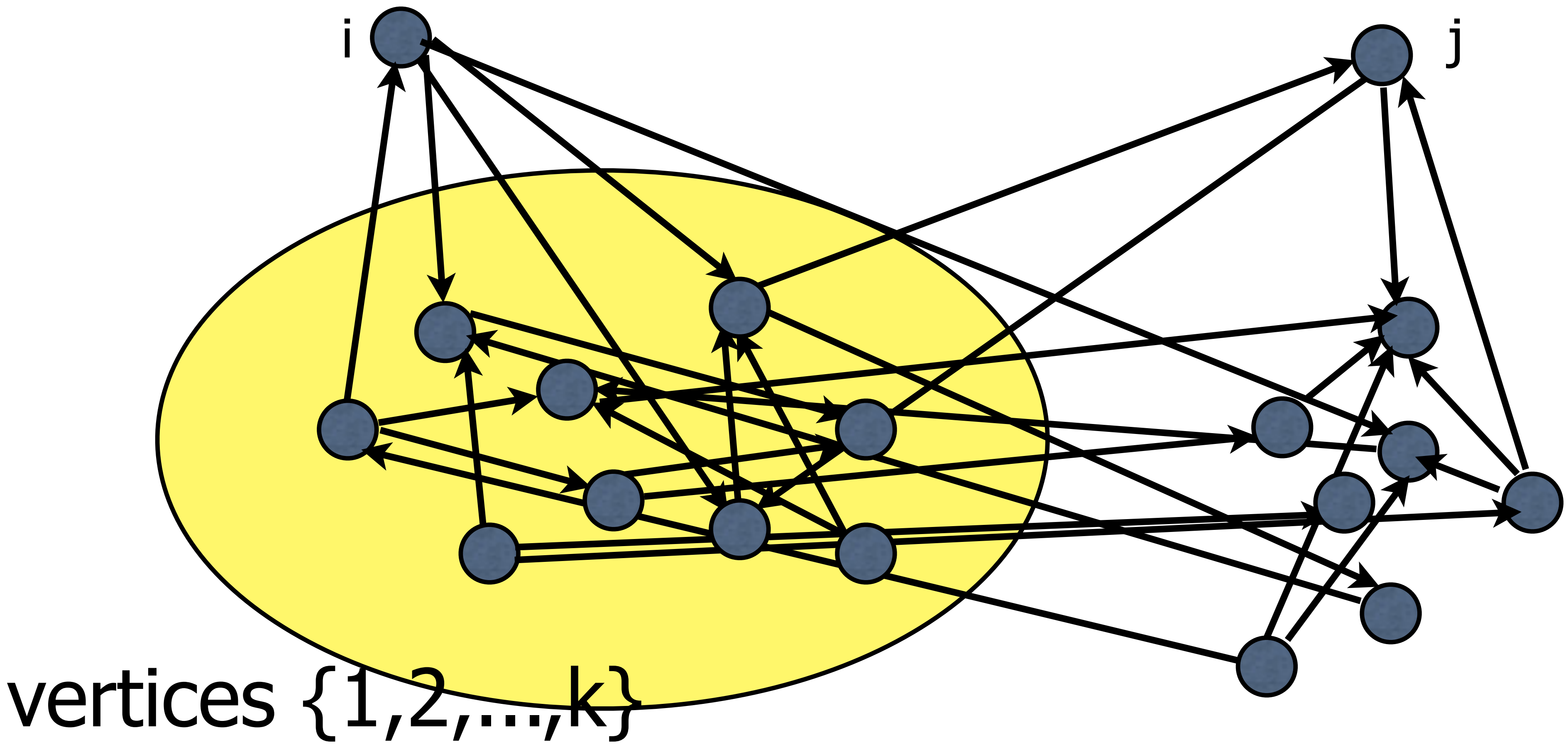
Given: directed graph, (possibly negative) edge weights

Goal: find shortest path between every two vertices

Subproblems: d(i,j,k) - length of shortest path that starts at i, ends at j and visits only {1,2,...,k} in the middle.

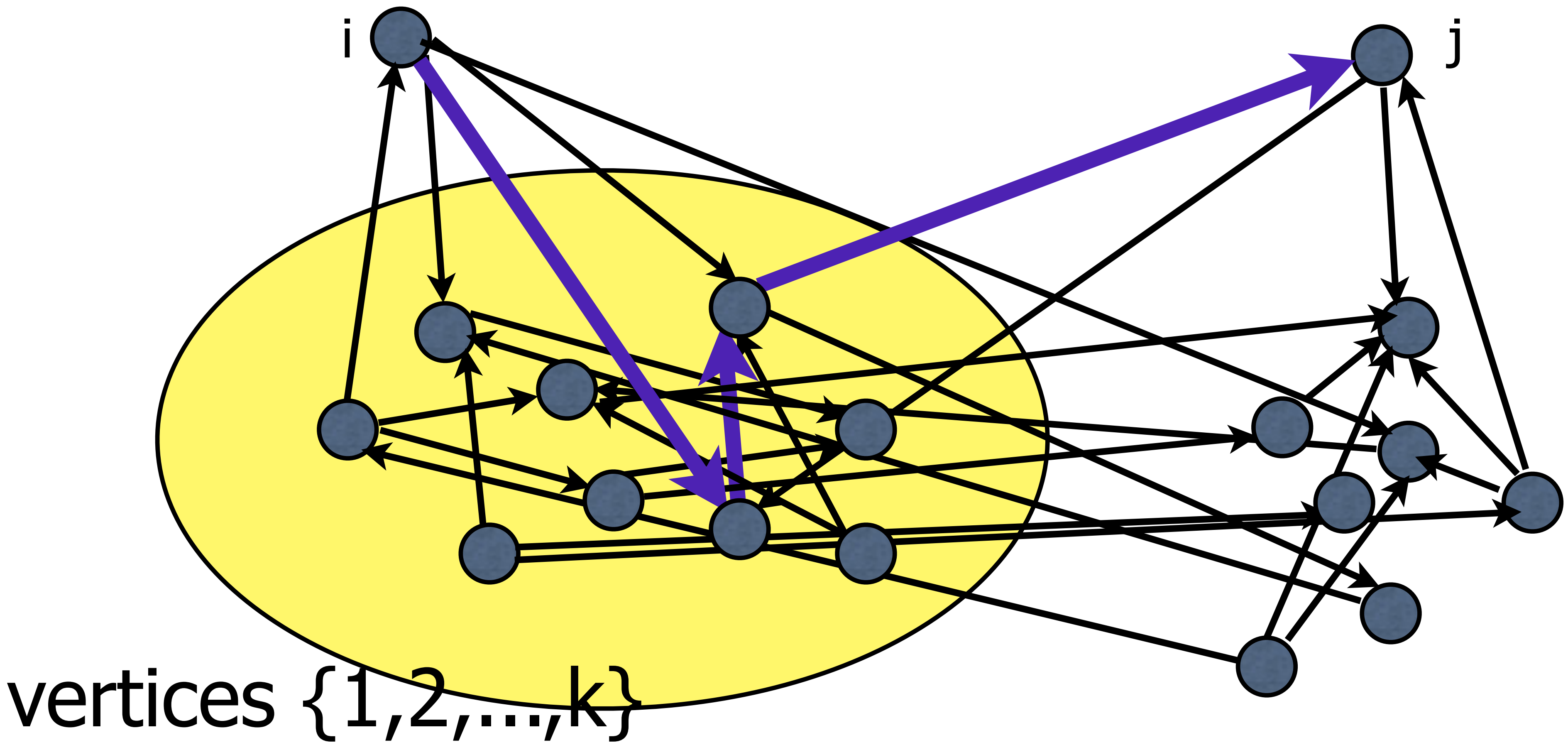
Goal: find shortest path between every two vertices

Subproblems: d(i,j,k) - length of shortest path that starts at i, ends at j and every other vertex on path is in {1,2,...,k}.



Goal: find shortest path between every two vertices

Subproblems: d(i,j,k) - length of shortest path that starts at i, ends at j and every other vertex on path is in {1,2,...,k}.

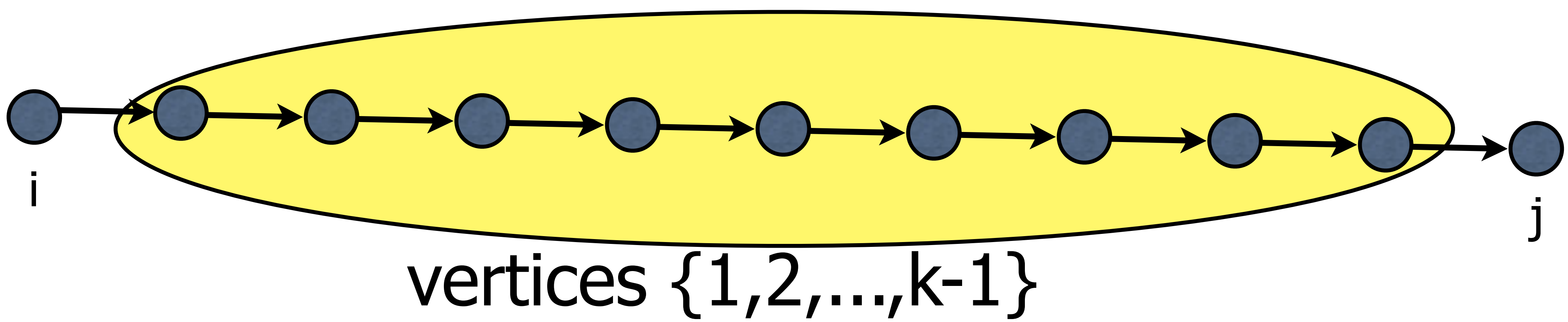


Subproblems: d(i,j,k) - length of shortest path that starts at i, ends at j and every other vertex on path is in {1,2,...,k}.

Observation:

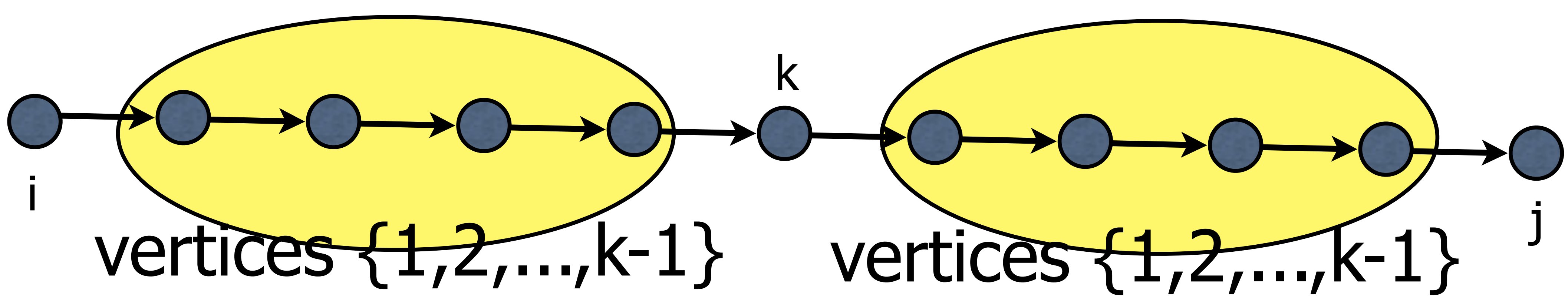
if shortest path for d(i,j,k) does not visit k, then

$$d(i,j,k) = d(i,j,k-1).$$



Otherwise,

$$d(i,j,k) = d(i,k,k-1) + d(k,j,k-1)$$



Subproblems: d(i,j,k) - length of shortest path that starts at i, ends at j and every other vertex on path is in {1,2,…,k}.

Claim: d(i,j,k) = min{d(i,j,k-1), d(i,k,k-1)+d(k,j,k-1)}

Algorithm:

```
for all i,j=1,…,n
    set d(i,j,0) = weight of edge (i,j)

for k=1,…,n
    for all i,j=1,…,n
        set d(i,j,k) = min{d(i,j,k-1),d(i,k,k-1)+d(k,j,k-1)}
```

<u>Running time</u> $O(n^3)$