CSE 417: Algorithms and Computational Complexity

Winter 2001

Lecture 13

Instructor: Paul Beame

1

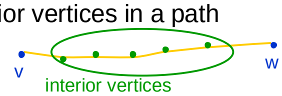
All-pairs shortest paths

- If no negative-weight edges and sparse graphs run Dijsktra from each vertex $O(nm \log n)$
- What about other cases?

2

Floyd's Algorithm Idea

- Interior vertices in a path



- Dijsktra's Algorithm
  - at each step always computed shortest paths that only had interior vertices from a set S at each step
- Floyd's Algorithm
  - slowly add interior vertices on fixed schedule rather than depending on the graph

3

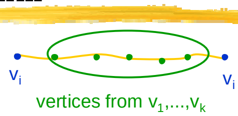
Floyd's Algorithm

- Vertices $V=\{v_1,\ldots,v_n\}$
- Let $D_k[i,j]$ = length of shortest path from v_i to v_j that only allows $\{v_1,\ldots,v_k\}$ as interior vertices
- Note
  - $D_0[i,j] = \begin{cases} \text{weight of edge } (v_i,v_j) & \text{if } (v_i,v_j)\in E \\ \infty & \text{if } (v_i,v_j)\notin E \end{cases}$
  - $D_n[i,j]$ = length of shortest path from v_i to v_j

4

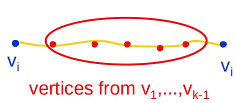
Floyd's Algorithm

Computing $D_k[i,j]$



Case 1: v_k not used

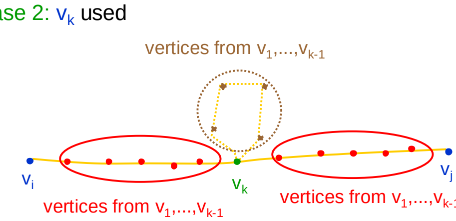
$D_k[i,j] = D_{k-1}[i,j]$



5

Floyd's Algorithm

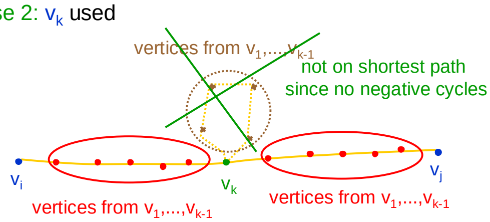
Case 2: v_k used



6

Floyd's Algorithm

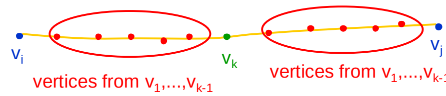
Case 2: v_k used



Floyd's Algorithm

Case 2: v_k used

$D_k[i,j] = D_{k-1}[i,k] + D_{k-1}[k,j]$



Floyd's Algorithm

$O(n^3)$ time
$O(n^2)$ storage by maintaining array for last two values of k

- Floyd(G)

```
D0←weight matrix of G
for k=1 to n do
    for i=1 to n do
        for i=1 to n do
            Dk[i,j]← min{ Dk-1[i,j],
                          Dk-1[i,k]+Dk-1[k,j]}
        endfor
    endfor
endfor
```


Multiplying Faster

- On the first HW you analyzed our usual algorithm for multiplying numbers
  - $\Theta(n^2)$ time
- We can do better!
  - We'll describe the basic ideas by multiplying polynomials rather than integers
  - Advantage is we don't get confused by worrying about carries at first

Note on Polynomials

- These are just formal sequences of coefficients so when we show something multiplied by x^k it just means shifted k places to the left

Polynomial Multiplication

- Given:
  - Degree n-1 polynomials P and Q
    - $P = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_2 x^2 + a_1 x + a_0$
    - $Q = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \ldots + b_2 x^2 + b_1 x + b_0$
- Compute:
  - Degree 2n-2 Polynomial P•Q
  - $P \bullet Q = a_{n-1}b_{n-1} x^{2n-2} + (a_{n-1}b_{n-2} + a_{n-2}b_{n-1}) x^{2n-3} + \ldots + (a_{n-1}b_{i+1}+a_{n-2}b_{i+2}+\ldots+a_{i+1}b_{n-1}) x^{n+i} + \ldots + a_0b_0$
- Obvious Algorithm:
  - Compute all a_ib_j and collect terms
  - $\Theta(n^2)$ time

Naive Divide and Conquer

- Assume n is a power of 2
  - $P = (a_{n-1}x^{n/2-1} + a_{n-2}x^{n/2-2} + \ldots + a_{n/2})\, x^{n/2}$
    $+ (a_{n/2-1}\, x^{n/2-1} + \ldots + a_2\, x^2 + a_1\, x + a_0)$
    $= P_1\, x^{n/2} + P_0$
  - $Q = Q_1\, x^{n/2} + Q_0$
- $P \bullet Q = (P_1 x^{n/2} + P_0)(Q_1 x^{n/2} + Q_0)$
  $= P_1 Q_1 x^n + (P_1 Q_0 + P_0 Q_1) x^{n/2} + P_0 Q_0$
- 4 sub-problems of size n/2 + plus linear combining
  - $T(n)=4T(n/2)+cn$
  - Solution $T(n) = O(n^2)$

13

Karatsuba's Algorithm

- A better way to compute the terms
  - Compute
    - $P_0 Q_0$
    - $P_1 Q_1$
    - $(P_1+P_0)(Q_1+Q_0)$ which is $P_1Q_1+P_1Q_0+P_0Q_1+P_0Q_0$
  - Then
    - $P_0Q_1+P_1Q_0 = (P_1+P_0)(Q_1+Q_0) - P_0Q_0 - P_1Q_1$
  - 3 sub-problems of size n/2 plus O(n) work
    - $T(n) = 3\, T(n/2) + cn$
    - $T(n) = O(n^{\alpha})$ where $\alpha = \log_2 3 = 1.59...$

14