CS 594: Representations in Algorithm Design **Spring 2022**

Lecture 17: 03/08/2022

Lecturer: Xiaorui Sun *Scribe: Chinmay Tarwate*

1 Last time and today

Previously: Graph representations and ideas and connections between different graph representations.
Today: Representations can be related to distribution, algebraic structure. why representations can be useful.

2 Representations

a. If there is an additional structure then it can give an additional property. This additional property can give more efficient algorithm. Each of this representation is a special case of general representation. Special cases have additional property. Examples:

- Tree: Tree is a special case of graph such that it does not have a cycle.
- Expander: In Edge expansion, for any cut of graph, there must exists one side of cut such that the volume is low.
- Low diameter graphs: Between 2 vertices there must exists a path that is not too long.

b. Solution for such special cases can be generalized for general cases of representations

- Approximation: Make use of special cases. Example: Using Low spanning tree as preconditioner to solve
- Decomposition: Partitioning graph into pieces with good properties and additional edges that are not good whose size is smaller than input size. Example: Low diameter decomposition, Expander decomposition. (¿ vertex partition)

c. Sparsification - Specific properties restrictions to reduce size of graph and not preserve entire graph

2.1 Decomposition

2.1.1 Short cycle decomposition

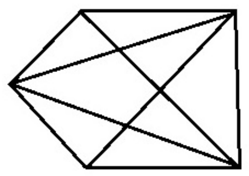
Definition: It is different in terms of vertex partition, It is about edge partition

$$E = E_1 U E_2 .....E_k U E_0$$

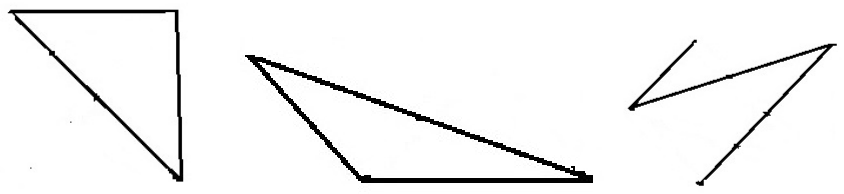
where E_1 to E_k are good properties and E_0 are not good properties

E_i is a short cycle and length is not too large
E_0 are bad edges and $|E_0|$ (size) is small

Arbitrary graph



Bad edges does not belong to cycle then take it out. Vertex can be present multiple times but edge appears only once.
Below are few examples of bad edges -



$E_1 ....E_0$ is a (α , β) short cycle decomposition where α, β are integers > 0

1. if each E_i such that i > 0 then E_i is a cycle of length $<= \alpha$
2. $|E_0| <= \beta$

If input is tree, then all edges of short cycle decomposition belong to E_0 and no E_1, E_2...

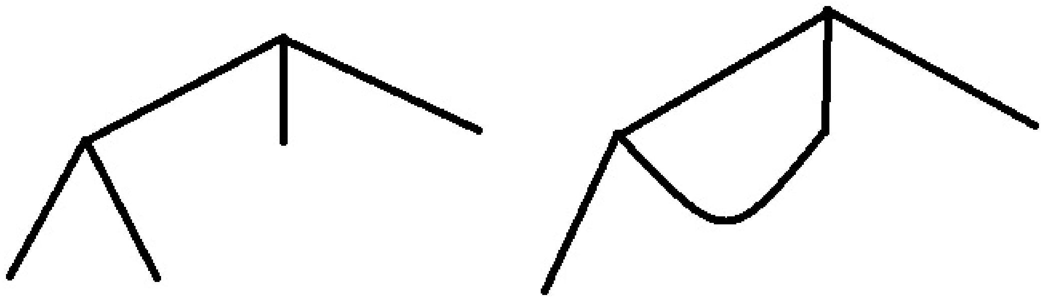
If graph is large cycle, then no short cycle decomposition as number of vertices close to number of edges

$\beta = O(nlogn)$ upper bounded by number of vertices(n)
α is as small as possible

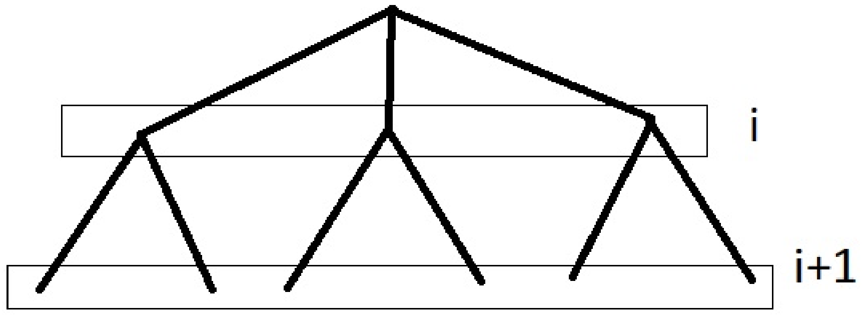
Theorem 1 *$\exists(O(logn), O(n))$ short cycle decomposition where $O(logn)$ is α and $O(n)$ is β*

Lemma 2 *If G(graph) has degree $>= 3$ for all vertices then G has a cycle of length $<= O(logn)$*

BFS from arbitrary vertex with atleast 3 branches



Other cases than above two have vertices as below and
Number of vertices in level i+1 >= 2∗number of vertices in level i -



2.1.2 Algorithm: Short cycle decompose

1. Make graph to be a graph with degree $>= 3$

2. Find a cycle of O(logn)
   Go to step 1 if graph is not empty

Iteratively put edges incident to vertices with degree 1 or 2 to E_0 until no vertex 1 or 2 is left
Put this cycle in short cycle decomposition
$\alpha = logn$ using lemma
$\beta = O(n)$

There exists algorithm components $(n^{O(1)}, n^{1+O(1)})$ short cycle decomposition in time $m^{1+O(1)}$ (where m is more number of bad edges)

3 Approximate Dynamic Programming (Edit distance for strings)

Edit distance Problem:
Input consists of 2 strings with respect to alphabet (ϵ)
Example: $\epsilon = C, T, A, G$, S_1 = CTACCG, S_2 = TACATG
Output: Minimum number of operations needed to transform S_1 to S_2
Operations :

1. Add one character at any location
2. Delete one character
3. Replace one character by another character

Example:
$S_1 -$ > CTACCG to $S_2 -$ > TACATG

Below is one way to transform -
Replace (1 position) C $-$ > T
$S_1 -$ > TTACCG
Replace (2 position) T $-$ > A
$S_1 -$ > TAACCG
Replace (3 position) A $-$ > C
$S_1 -$ > TACCCG
Replace (4 position) C $-$ > A

$S_1 -> $ TACACG
Replace (5 position) C $->$ T
$S_1 -> $ TACATG

Number of operations: 5

Better way to transform will be with less number of operation as follows -
Delete (1 position) C
$S_1 -> $ TACCG
Replace (4 position) C $->$ A
$S_1 -> $ TACAG
Add (5 position) T
$S_1 -> $ TACATG

Number of operations: 3
Time complexity of these transforms will be $O(n^2)$

3.1 Standard dynamic programming solution

d(i,j) = Minimum number of operations required to transform the first i characters of S_1 to first j characters of S_2 where d is the edit distance

Example: d(2,3) $->$ Number of operations to transform CT to TAC: 3
Edit distance = d($|S_1|, |S_2|$)
Problem: How to determine d(i,j)
d(i,j) = 0 if i = j = 0
d(i,j) = i if j = 0
d(i,j) = j if i = 0
d(i,j) = (replaced/deleted/not replaced or deleted) if i != 0 and j != 0

4 Next time

Solve the edit distance problem to complexity of $O(n^{1.6..})$