1 Formeln

$G = (V, E)$ $n = |V|$ $m = |E|$
$\Omega(n)$
$\Theta(n)$
$scan(N) = \Theta(N/B)$
$sort(N) = \Theta((N/B) \cdot \log_{M/B}(N/B))$
$scan(N) < sort(N) \ll N$
$\mathcal{O}(|V|)$
$\mathcal{O}((1 + \log\log(B \cdot n/m)) \cdot sort(n + m))$
$\mathcal{O}(n/\mu)$
$\mu = \sqrt{B}$
$G_{i-1}(d_{i-1})$ $G_i(d_i)$
$\max\{0, d_{i-1}(v) - \alpha\}$
$\mathcal{O}(n/B^{2/3} + sort(n) \cdot \log(B))$
$d_{i-1}(v) - d_i(v) \leq \alpha$
$d_{i-1}(v) - d_i(v) > \alpha$
$\mathbf{P}[r(v) = 0] = \mathbf{P}[r(v) = 1] = \frac{1}{2}$
$\mu = 2^1, 2^2, \ldots, \sqrt{B}$
$\langle b_r, \ldots, b_{q+1}, b_q, \ldots, b_1 \rangle$
prefix $\langle b_r, \ldots, b_{q+1} \rangle$ suffix $\langle b_q, \ldots, b_1 \rangle$
External-Memory (EM)

2 Paper 2

2.1 Introduction

Was ist BFS, edges one-by-one insertion. output nach jeder insertion.
scheint unnötig auf dünn besetzten Graphen (da $\Omega(n)$) => dem ist nicht so.
BFS level computation »> reporting them. => schnelle alternative.

2.2 I/O-Model and Related Work

2.2.1 Computation model

two-level hierarchy: fast internal memory (store M data items), slow disk of infinite size. External-Memory (EM) model of Aggarwal and Vitter [1].
In an I/O operation, one block of data, which can store B vertices/edges, is transferred between disk and internal memory. The measure of performance of an algorithm is the number of I/Os it performs. The number of I/Os needed to read N contiguous items from disk is scan(N)=$\Omega(N/B)$. The number of I/Os required to sort N items is sort(N)=$\Omega((N/B)log_{M/B}(N/B))$. For all realistic values of N, B, and M, $scan(N) < sort(N) << N$.

2.2.2 Review of Static and Dynamic EM BFS Algorithms

MM_BFS algo and its dynamic extension. Edge insertions on already connected undirected sparse graphs with $n = |V|$ and $m = |E| = \mathcal{O}(|V|)$ edges.

2.2.3 MM$_B FS$

Zwei Phasen: preprocessing und BFS phase.
Preprocessing: algo produces a clustering. => kann über euler-tour based on arbitrary spanning tree T des Graphen G realisiert werden. Falls T nicht im Input inbegriffen ist, kann T durch $\mathcal{O}((1 + \log\log(B \cdot n/m)) \cdot sort(n + m))$ I/O's generiert werden.
Jede ungerichtete Kante wird durch zwei gerichtete ersetzt. Jeder Knoten wählt eine zyklische ORDER seiner Nachbarn. Successor of incoming edge is defined to be the outgoing edge of the next node in the cyclic order. Die Tour wird an einem speziellen Knoten (zb Wurzel) unterbrochen, wobei die Elemente der resultierenden Liste in CONSECUTIVE order using an external memory list-ranking algorithm.
Danach wird die Euler Tour in Cluster von μ Knoten geteilt und DUPLICATES werden entfernt, so dass sich ein Knoten nur im ersten Cluster befindet, in dem er zuerst auftritt. Dies bedarf einige Sortierschritte.
Somit ist der maximale Abstand von zwei Knoten innerhalb eines Clusters durch $\mu - 1$ beschränkt, die Größe der Cluster selbst durch $\mathcal{O}(n/\mu)$. In der BFS Phase, die Grundidee ist die prerpoccessed Cluster in eine effiziente Datenstruktur (hot pool) zu laden (kostet ein paar I/O's), da die Cluster vertices enthalten, die in benachbarte BFS level übergehen.
Somit können benachbarte Knoten $N(l)$ eines BFS level l allein durch scannen des hot pools berechnet werden. Das nächste BFS level wird durch entfernen der besuchten Knoten in Level $l - 1$ und l von $N(l)$. Jedoch können neu entdeckte Nachbarn zu bisher unbesuchten Clustern gehören. Unstrukturierte I/O's sind nötig, um die Cluster in den hot pool zu laden, von wo sie wiederum entfernt werden, sobald die entsprechenden BFS level erzeugt wurden.

MISSING

2.2.4 Dynamic BFS

Review the high-level ideas to computing BFS on general undirected sparse graphs in an incremental setting. Insertion of the ith edge (u, v) und graph before: $G_{i-1}(d_{i-1})$, graph danach: $G_i(d_i)$.
Run BFS phase of $MM_B FS$. Unterschied: adj list für v is added to the hot pool H when creating BFS level max blub of G_i für ein $\alpha > 1$.
Indem man die adj list sortiert hält bzgl der Entfernung der node distances in G_{i-1} kann man dies für alle Nodes v in I/O's $d_{i-1}(v) - d_i(v) \leq \alpha$ erreichen.

3 On Dynamic Breadth-First search in External-Memory

3.1 Introduction

BFS: Decomposes Graph $G(V, E)$ of n nodes and m edges into at most n levels.
Objective $\Rightarrow$ efficiently process an online sequence of update and query operations.

$\Omega(n)$ edge insertions, but not delitions.

3.1.1 Computiation models

External-Memory (EM) model of Aggarwal and Vitter [1].
In an I/O operation, one block of data, which can store B vertices/edges, is transferred between disk and internal memory. The measure of performance of an algorithm is the number

of I/Os it performs. The number of I/Os needed to read N contiguous items from disk is scan(N)=$\Omega(N/B)$. The number of I/Os required to sort N items is sort(N)=$\Omega((N/B)log_{M/B}(N/B))$. For all realistic values of N, B, and M, $scan(N) < sort(N) << N$.

3.1.2 Results

For general sparse undirected graphs of initially n nodes