UNIVERSIDADE FEDERAL DO CEARÁ

Equipe: José Robertty e Maria Tassiane
Disciplina: Estrutura de Dados Avançada
Professor: Fábio Dias

Este trabalho tem como objetivo observar algumas variações nas implementações de conjuntos disjuntos para o algoritmo Kruskal, com foco na analisar do custo computacional.

Especificações da Máquina:

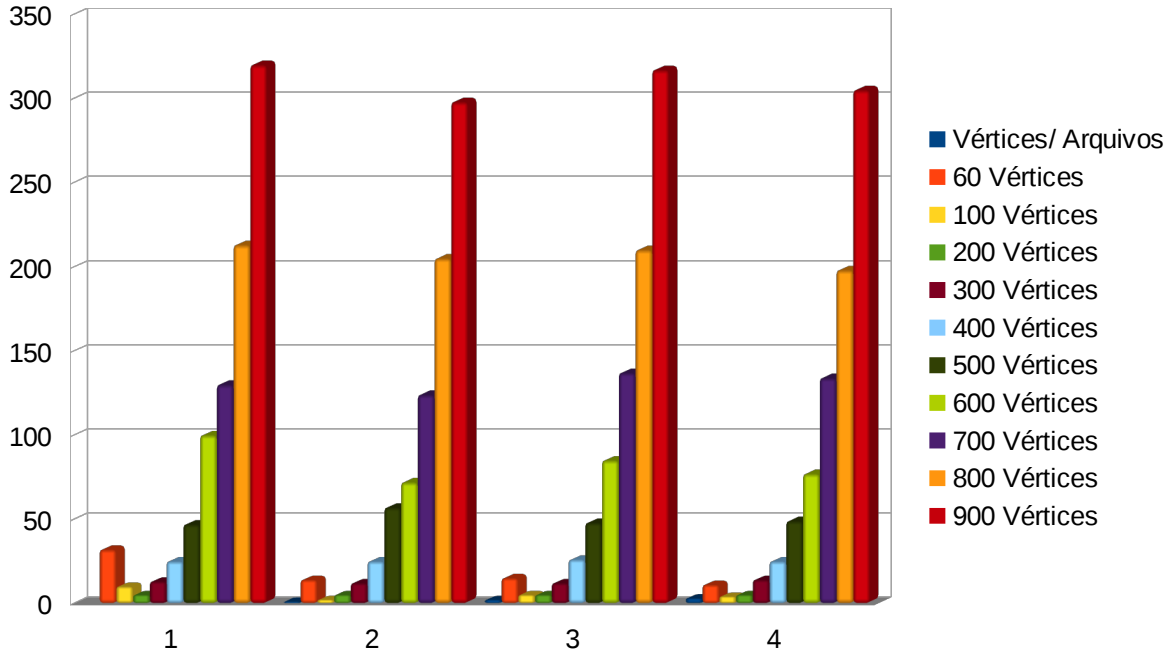
- Sistema operacional: Ubuntu 15.10
- Processador: Intel core i5-4210U 1.70GHz
- Memória: 7,7 GiB
- Tipo de sistema: 64-bit

A implementação utilizada para Conjuntos Disjuntos neste trabalho é conhecida como “Floresta de conjuntos disjuntos” e possui como principais funções: Make_set, Find_set, Union e Link. Nele iremos comparar os tempos deste algoritmo com variações de implementação das funções Find_set e Link.

Iremos agora apresentar as tabelas de tempos (todos os tempos em milissegundos):

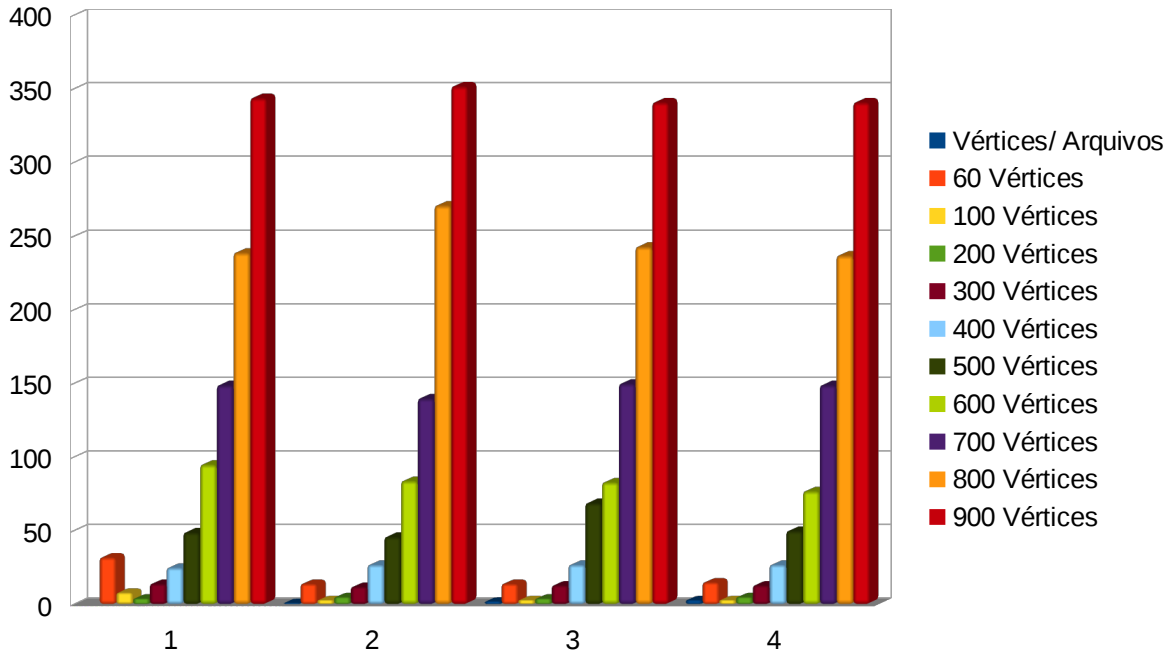
Vértices/ Arquivos	**60**	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**
0	32	10	5	13	25	47	100	130	213	320
1	14	2	5	12	25	57	72	124	205	298
2	15	5	5	12	26	48	85	137	210	317
3	11	4	5	14	25	49	77	134	198	305

Tabela 1 – Tempos do código implementado com FIND_SET bom e LINK bom.



Vértices/ Arquivos	60	100	200	300	400	500	600	700	800	900
0	32	8	4	14	25	49	95	149	239	344
1	14	3	5	12	27	46	84	140	271	352
2	14	3	4	13	27	69	83	150	243	341
3	15	3	5	13	27	50	77	149	237	341

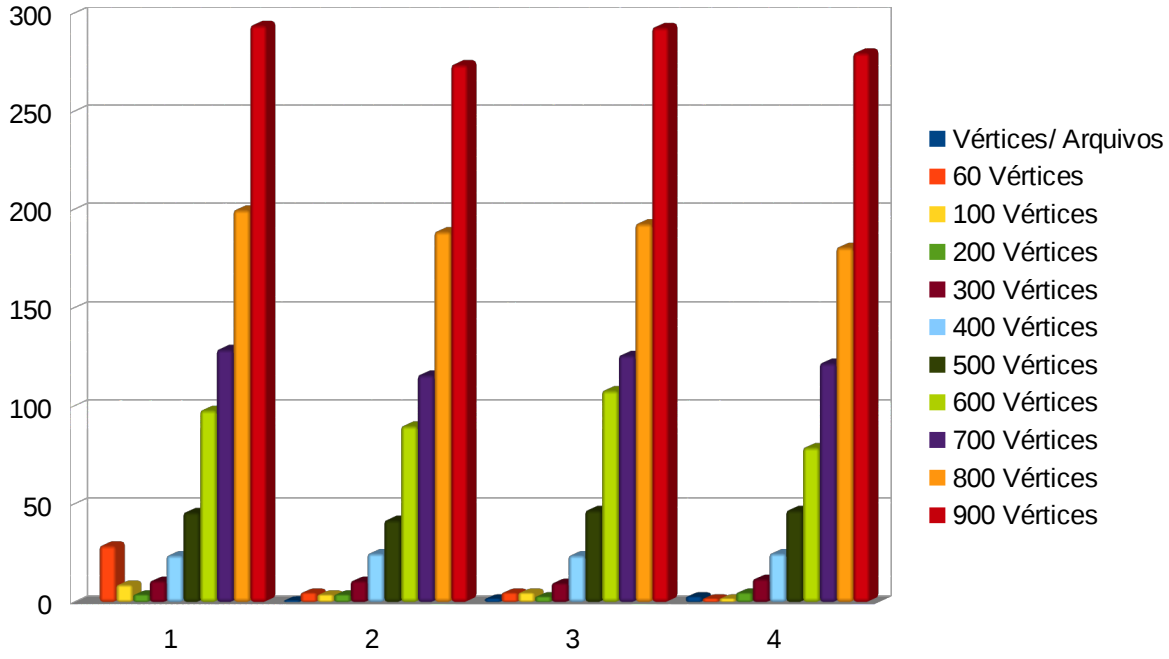
Tabela 2 – Tempos do código implementado com FIND_SET bom e LINK ruim.



Podemos já aqui fazer a primeira comparação, pois da tabela 1 para a tabela 2 só o LINK que variou, podemos observar que com a implementação do LINK ruim a execução é mais custosa, sobretudo quando o número de vértices aumenta. Em 30 dos 36 tempos a implementação com o LINK ruim possui tempo maiores ou iguais a implementação usando o LINK bom.

Vértices/ Arquivos	60	100	200	300	400	500	600	700	800	900
0	29	9	4	11	24	46	98	129	200	294
1	5	4	4	11	25	42	90	116	189	274
2	5	5	3	10	24	47	108	126	193	293
3	2	2	5	12	25	47	79	122	181	280

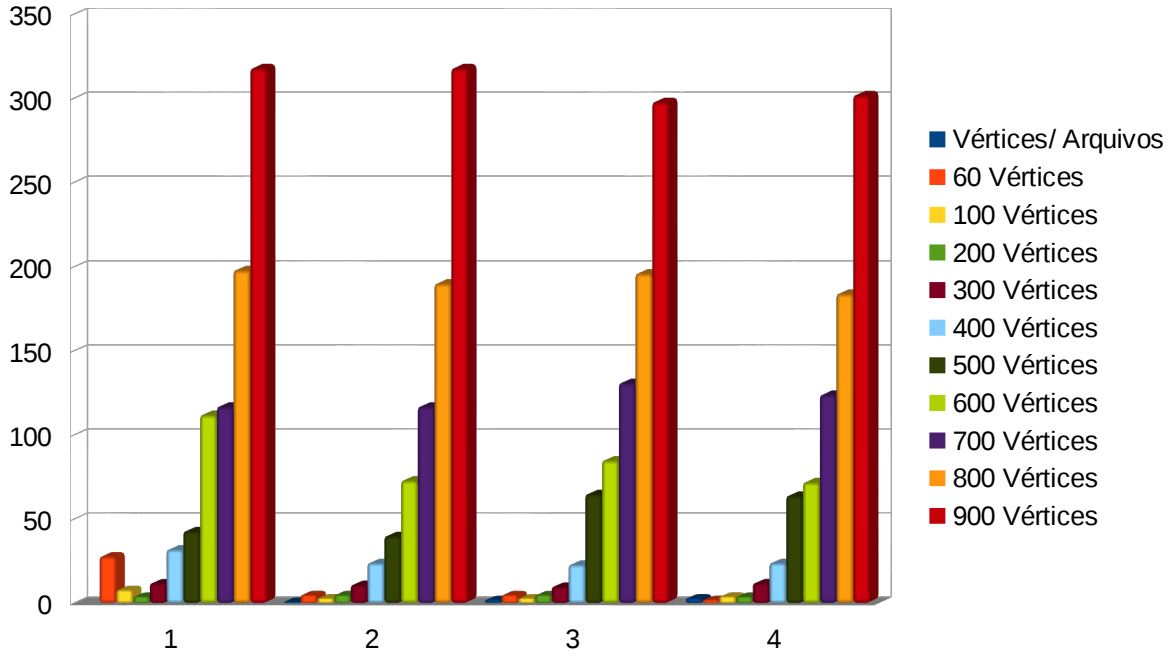
Tabela 3 – Tempos do código implementado com FIND_SET ruim e LINK bom.



Aqui cabe a comparação da tabela 1 com a tabela 3, pois nessas tabelas só a implementação do FIND_SET é alterada. Aqui percebemos algo que nos levou a um estranhamento, pois a implementação da tabela 3 é sempre mais rápida em 28 dos 36 tempos obtidos do que a implementação da tabela 1, ou seja, o FIND_SET bom aparentemente não é tão bom assim.

Vértices/ Arquivos	**60**	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**
0	28	8	4	12	32	43	112	117	198	318
1	5	3	5	11	24	40	73	117	190	318
2	5	3	5	10	23	65	85	131	196	298
3	2	4	4	12	24	64	72	124	184	302

Tabela 4 – Tempos do código implementado com FIND_SET ruim e LINK ruim.



Podemos comparar a tabela 3 e a tabela 4, pois o FIND_SET manteve-se constante e só que variou foi o LINK. Em 24 dos 36 tempos a implementação com LINK ruim teve tempo maior ou igual a implementação do LINK bom. Relembre-se que na nossa primeira comparação de tabelas o LINK bom também foi superior ao LINK ruim, portanto o LINK bom é realmente a melhor implementação.

A última comparação que podemos fazer é com a tabela 2 e a tabela 4, onde a implementação do FIND_SET variou. Em somente em 15 dos 36 casos o tempo do FIND_SET bom foi melhor do que o FIND_SET ruim. Porém a maioria desses casos com um número de Vértices maiores, o que nos diz que o FIND_SET bom só será superior quando o número de vértices aumentar mais.