MANUAL
DO
PROGRAMADOR

Natal/RN

2014

Analisando o problema, percebemos que para facilitar o desenvolvimento do programa, precisávamos subdividir o problema. Utilizando técnicas para análise de projeto, mapeamos as seguintes classes: casa, conexão, cidade, main e Union-Find.

Casa: simboliza uma casa com sua identificação e suas conexões.

Conexão: representa a conexão entre duas casas bem como o respectivo valor dessa conexão.

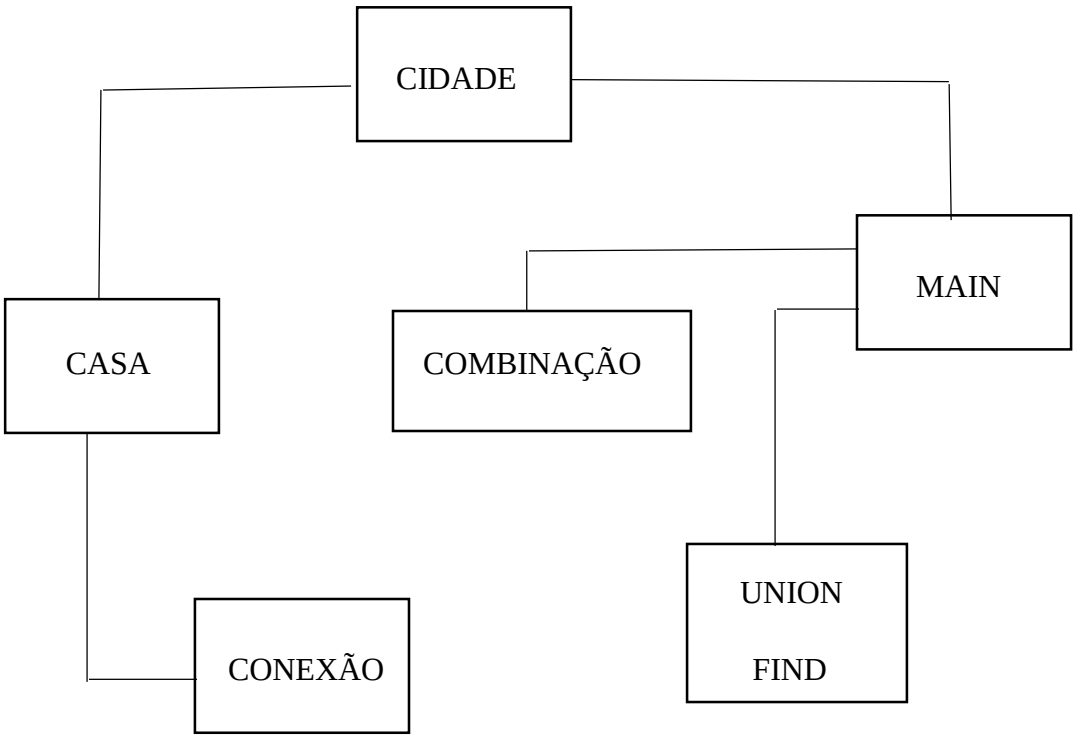
Cidade: contém todas as casas.

Main: responsável pela leitura de arquivos.

Union-Find: classe auxiliar.

Combinação: responsável por gerar as combinações.

O diagrama de classes representando todas as classes existentes no programa segue:



A cerca das estruturas de dados não-triviais utilizadas, fizemos uso dos Conjuntos Disjuntos. A nossa implementação, assim como na literatura, foi feita com floresta de árvores e utilizando as heurísticas de union by rank e path compression. Além dos métodos clássicos para manipulação de Conjuntos Disjuntos (makeSet, Find e Union) nós ainda implementamos um método para verificar quando o representante de dois conjuntos é o mesmo. Nesse trabalho, utilizamos os conjuntos disjuntos para garantir o formato da rede como uma árvore, ou seja, excluindo os ciclos.

A complexidade das classes casa, conexão, cidade, main, union-find e combinação, são respectivamente: $\Theta(1)$, $\Theta(1)$, $\Theta(1)$, $O(n^4)$, $O(m)$ e $O(n* 2^n)$. Como a complexidade do programa é dada pela instrução (ou conjunto de instruções) que é executada mais vezes, assim vamos explicar como obtemos a complexidade de maior grau, $O(n* 2^n)$.

Essa complexidade é dada devido a operação de achar todas as combinações de soluções possíveis. Note que, para uma cidade com N casas podemos ter no máximo 2^n combinações de ligações possíveis, assim sendo, nosso algoritmo irá efetuar 2^n operações, o que nos dá uma complexidade de $O(2^n)$. Contudo, nosso algoritmo se vale de alguns artifícios tais como obter o tamanho de uma String para achar uma combinação, tal operação leva $O(n)$ para ser executada. Concluímos portanto, que a complexidade geral para se achar todas as combinações possíveis utilizando o nosso algoritmo é $O(n* 2^n)$. Como já foi dito, essa é a instrução que é executada mais vezes no nosso projeto, logo ela indicará a complexidade geral do programa, portanto $O(n*2^n)$.