```
}

/* Treap Functions */
Treap
tree_treap_create(idx_t initial_capacity) {
   Treap new_treap;
   new_treap.nodes = (TreapNode*) malloc(sizeof(TreapNode) * initial_capacity);
   if (new_treap.nodes == NULL) {
      perror("Failed to allocate Treap.");
      exit(EXIT_FAILURE);
   }

   new_treap.tree_root = IDX_INVALID;
   new_treap.elements = 0;
   new_treap.capacity = initial_capacity;

   /* inicializar novos nós */
   TreapNode* endptr = new_treap.nodes + initial_capacity;
   for (TreapNode *ptr = new_treap.nodes; ptr != endptr; ptr++) {
      *ptr = (TreapNode){0, 0, IDX_INVALID, IDX_INVALID};
   }

   return new_treap;
}

void tree_treap_resize(Treap *treap) {
   /* se a capacity for máxima */
   if (treap->capacity == IDX_INVALID - 1) return;

   idx_t old_capacity = treap->capacity;
   idx_t new_capacity = (idx_t)(treap->capacity * RESIZE_FACTOR);

   /* se a nova capacity for maior que a capacidade máxima */
   if (new_capacity < treap->capacity) {
      new_capacity = IDX_INVALID - 1;
   }

   TreapNode *new_nodes = (TreapNode*) realloc(treap->nodes, sizeof(TreapNode) * new_capacity);
   if (new_nodes == NULL) {
      perror("Failed to realloc new nodes.");
      exit(EXIT_FAILURE);
   }
   treap->nodes = new_nodes;

   /* incializar nova memóra */
   for (idx_t i = old_capacity; i < new_capacity; i++) {
      treap->nodes[i] = (TreapNode){0, 0, IDX_INVALID, IDX_INVALID};
   }

   treap->capacity = new_capacity;
}

void
tree_treap_destroy(Treap *treap) {
   free(treap->nodes);
   treap->capacity = 0;
   treap->elements = 0;
}


static idx_t
```