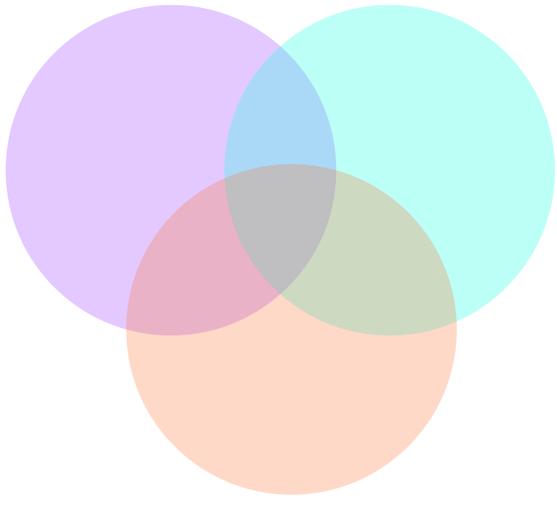
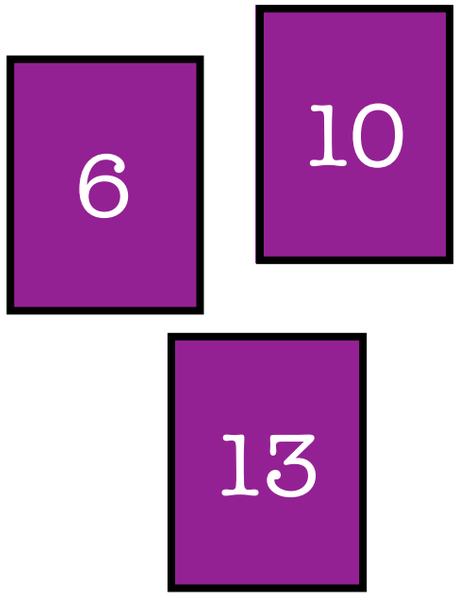
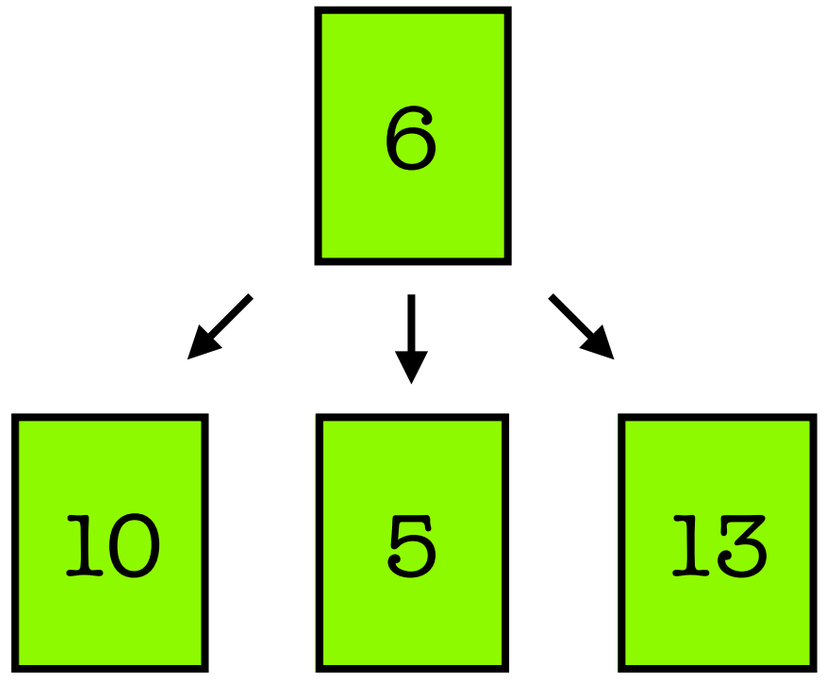
containers

(estruturas & TADs)



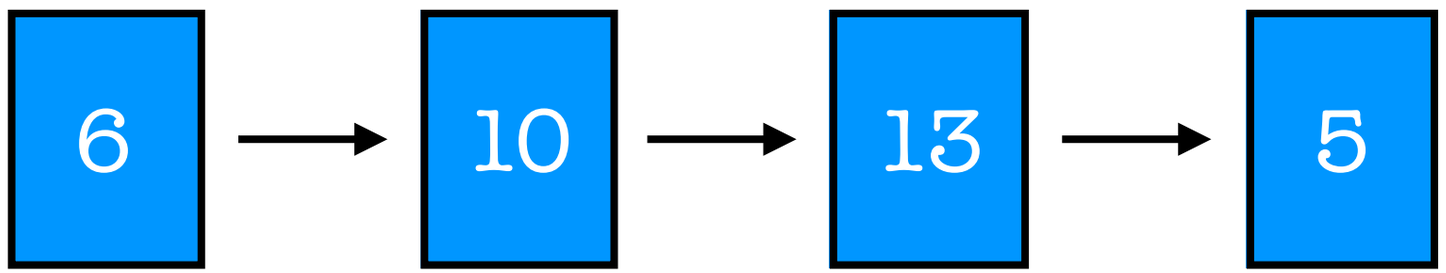


(não-relacionais)



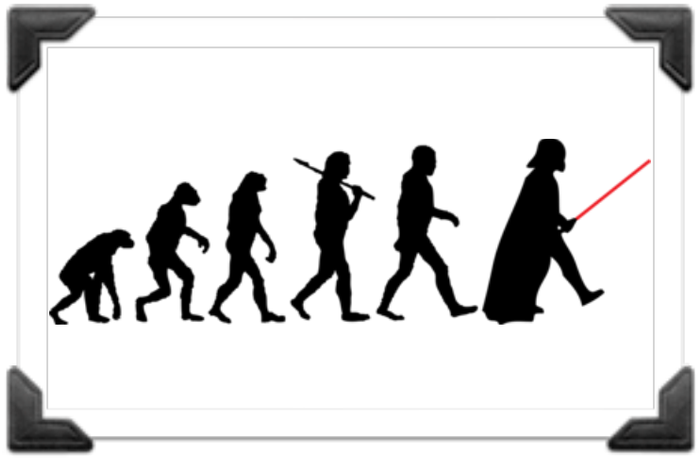
(hierárquicos)

TADs



(lineares)

lista



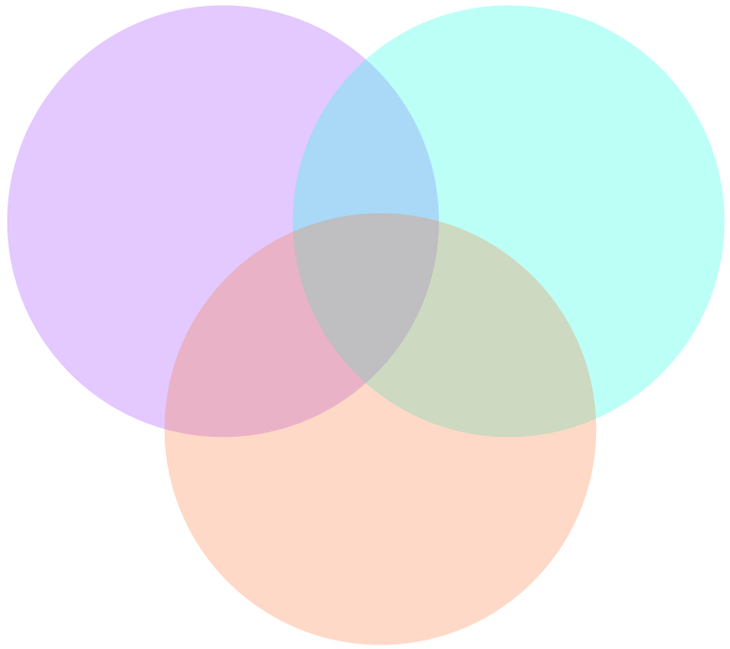
sequência

lista

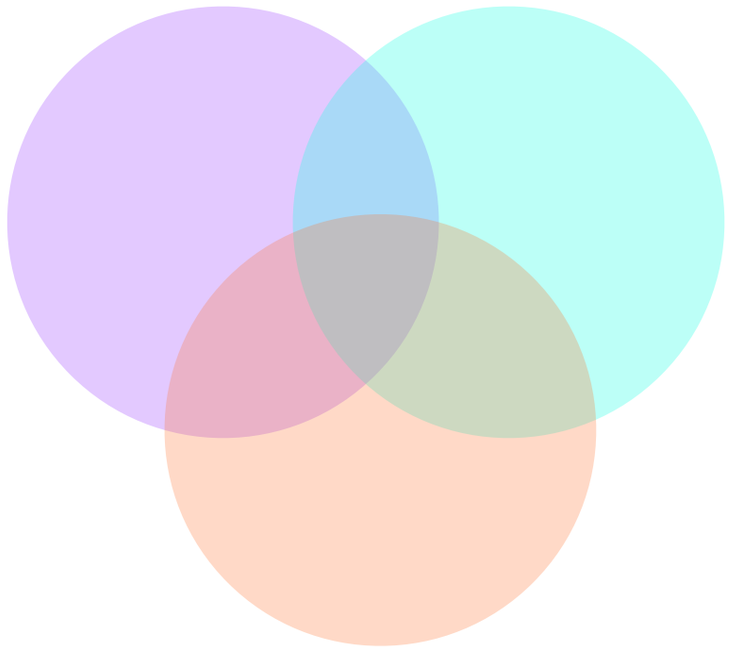
deque

pilha

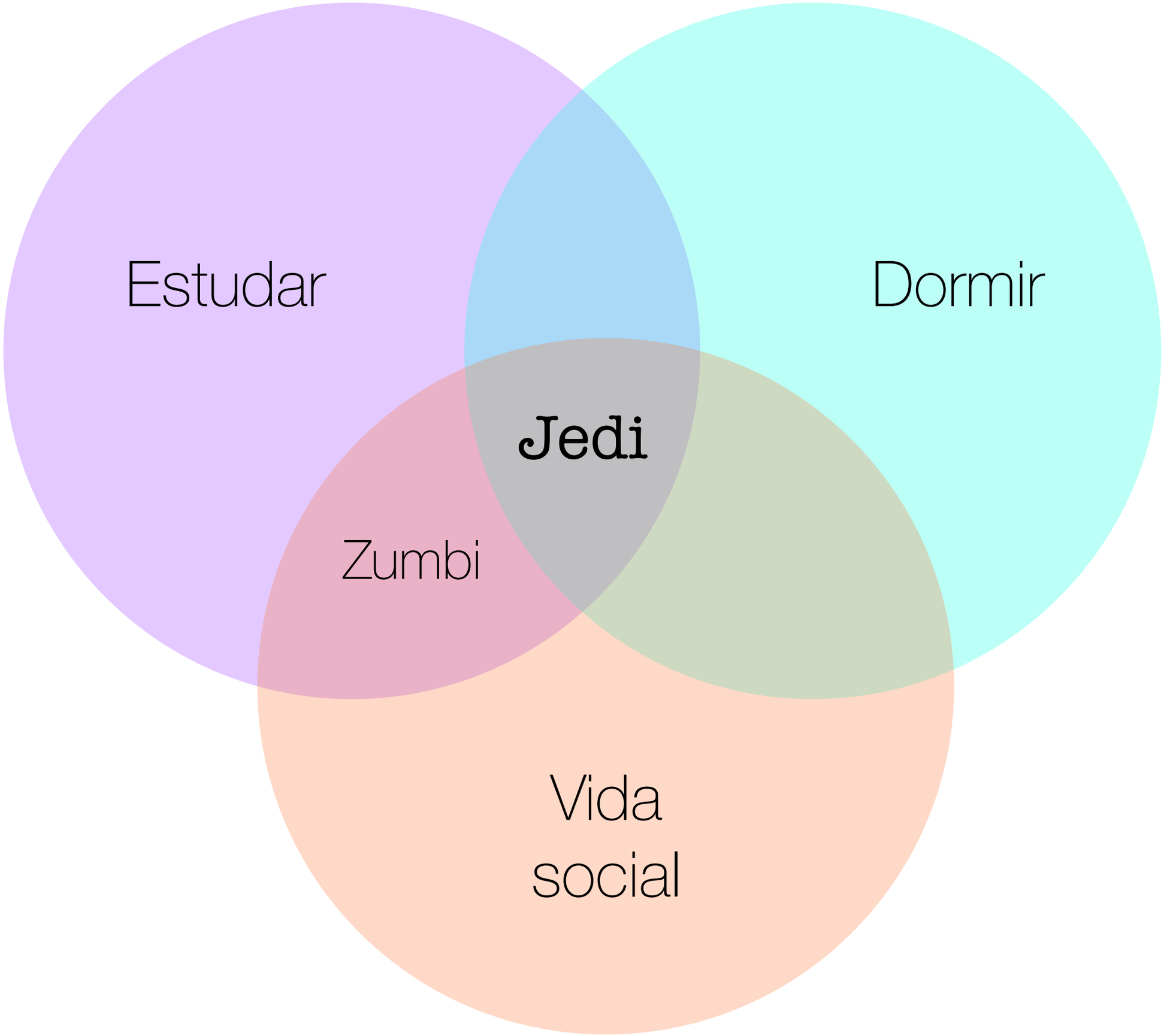
fila



TADs | (não-relacionais)

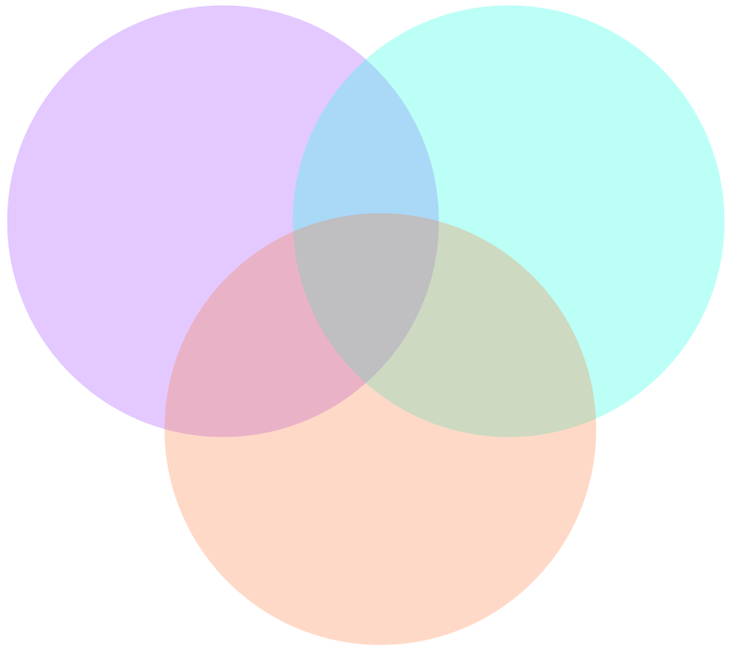


TAD | (conjunto)

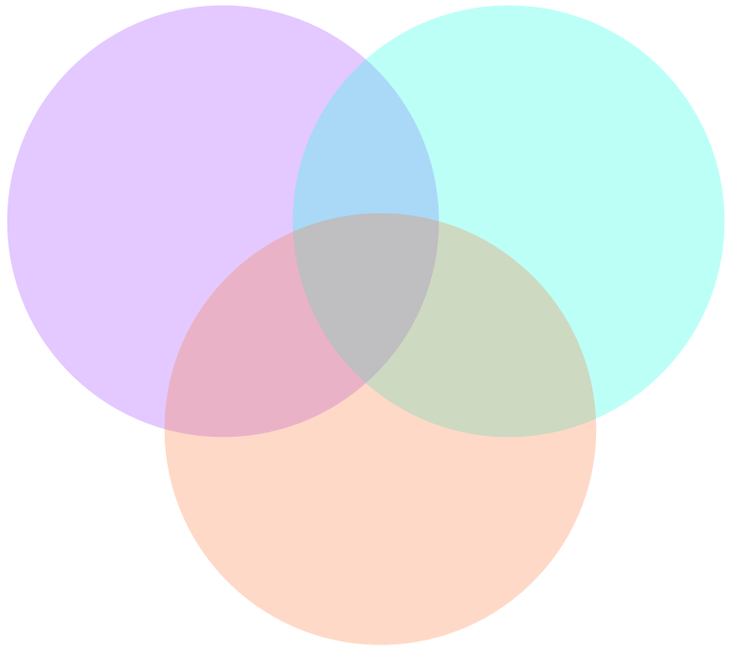

Estudar
Dormir
Jedi
Zumbi
Vida
social

TADs | (dicionários)

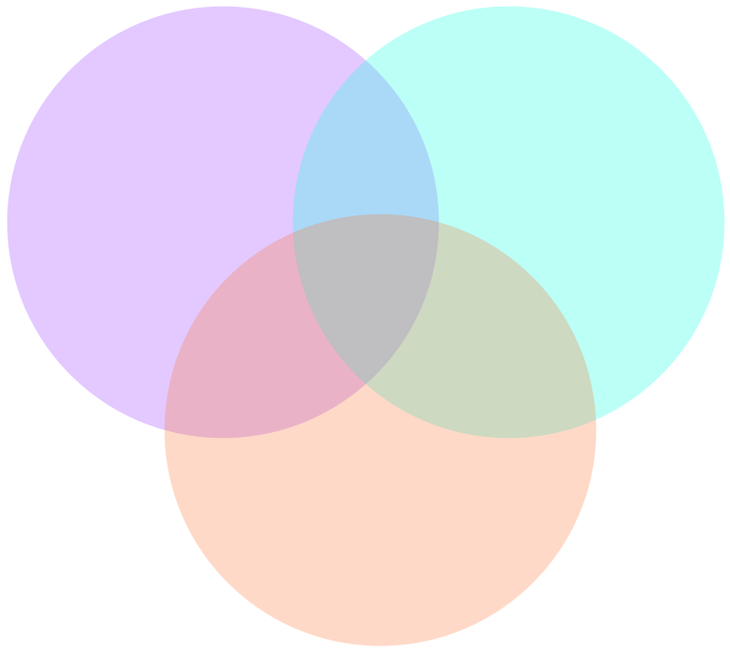

mag·ni·fier
that magnifies; spe
ing
mag·nify (mag'nə fī')
fier < L. magnificare, to
worship < magnus, grea
{Rare} to make greater in
cause to seem greater, mo
gerate {to magnify one's suf
really so; increase the appa
lenses 4 {Archaic} to glorif
increasing the apparent size
magnifying glass a lens or
apparent size of an object
mag·nil·o·quent (mag nil'ə
quence < L magniloquent
< magnus, great (see MA
or grandiose in speech or
—mag·nil'o·quence n.
Mag·ni·to·gorsk (mäg
River; pop. 422,000
mag·ni·tude (mag
great: see MAGN
importance
able qua
of the first magnitude of the greatest impor
mag·no·lia (mag nōl'yə, -nōl'ē-) n. [Mod.L., after P. Magnol (1638-
(Magnolia) of trees or shrubs of the
magnolia family, with large, fragrant
flowers of white, pink, or purple 2 the
—adj. designating a family
order Magnoliales) of



TADs | (não-relacionais)



adicionar | remover | pertinência



união

intersecção

diferença

contido

adicionar

remover

pertinência

acesso

adicionar | remover | pertinência

Operação	TAD Conjunto / Dicionário			
	Vetor circular		Lista encadeada	
	Melhor	Pior	Melhor	Pior
adicionar	Θ(1)	Θ(1)	Θ(1)	Θ(1)
remover	Θ(1)	Θ(n)	Θ(1)	Θ(n)
pertinência	Θ(1)	Θ(n)	Θ(1)	Θ(n)

Tabela de
Dispersão

Encadeamento
Simples

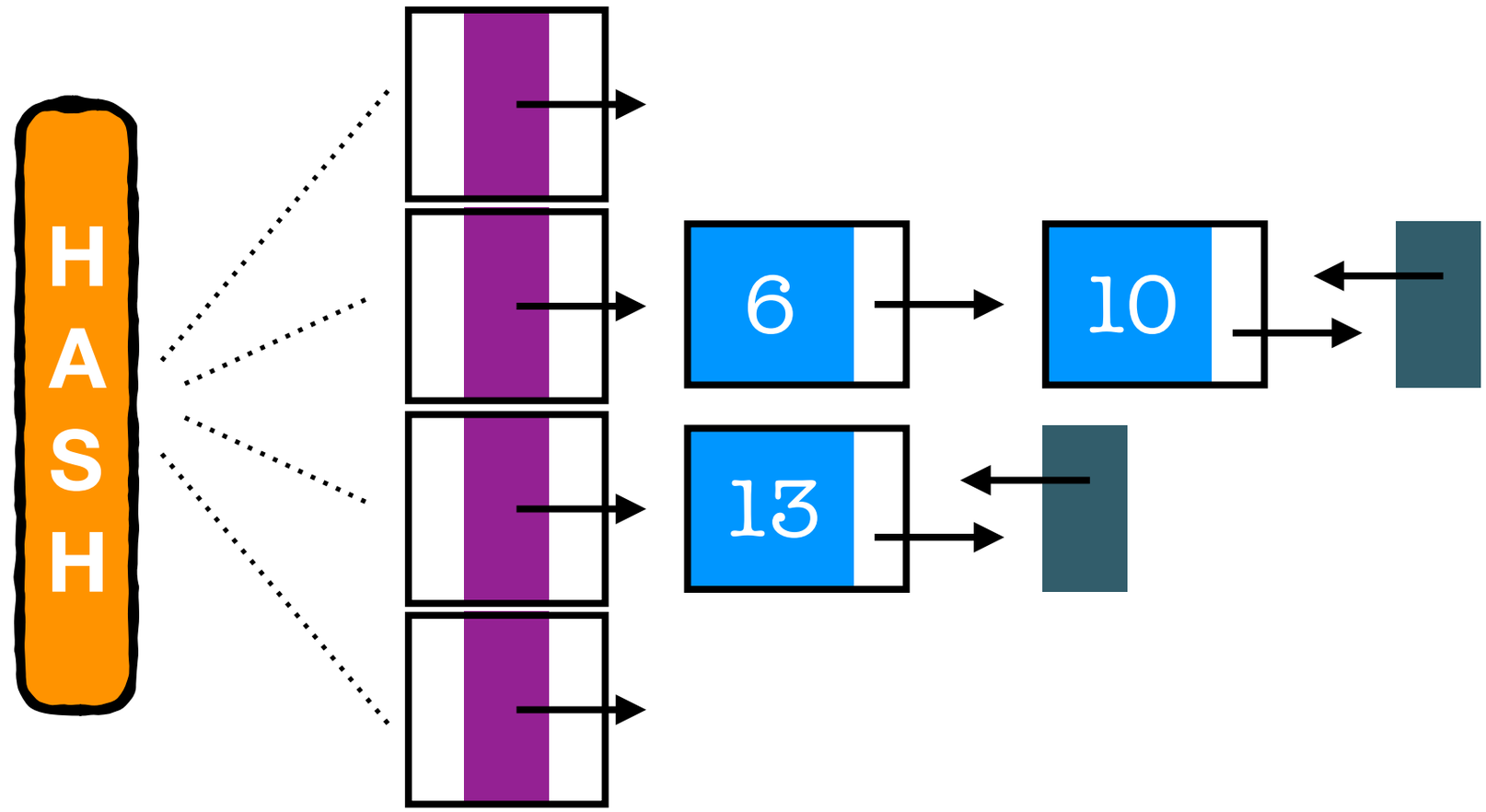
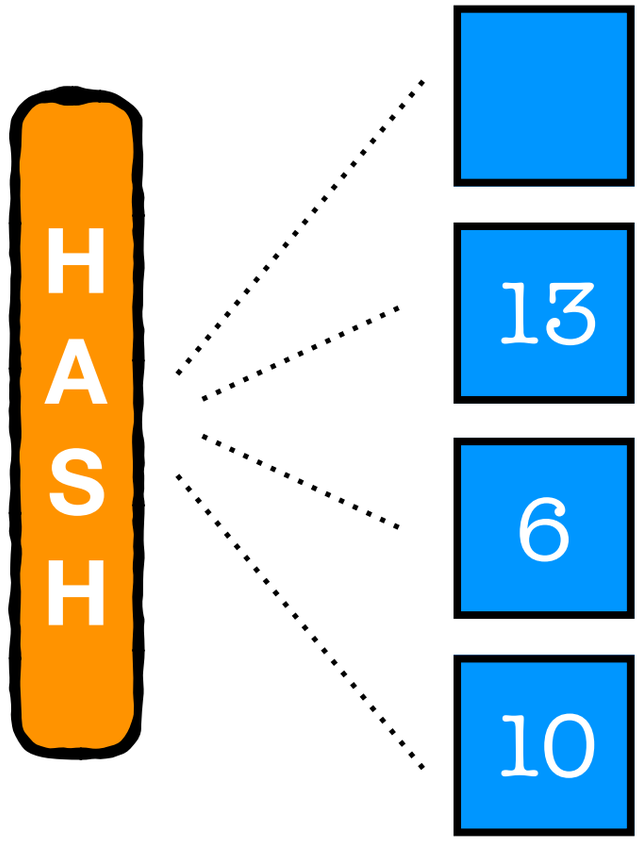
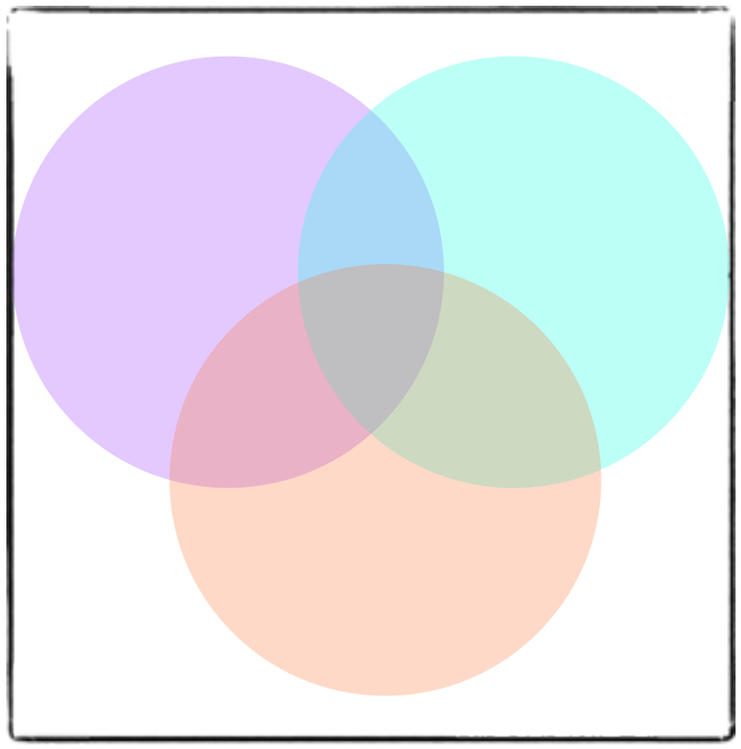


Tabela de
Dispersão

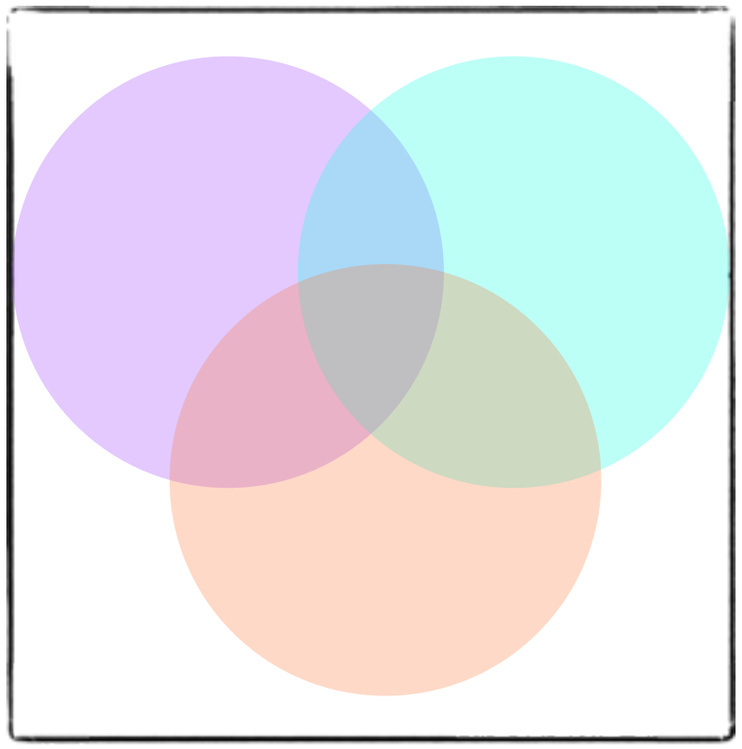


Endereçamento
Aberto



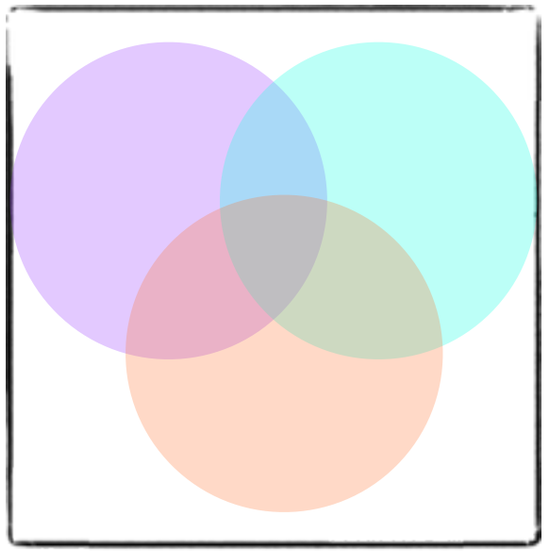
conjuntos

```
template <typename T>
class Set {
    private:
        chainedHashTable<T> data;
    public:
        unsigned size (void) {
            return data.size();
        }
        bool find (const T & value) {
            return data.find(value);
        }
        void add (const T & value) {
            data.add(value);
        }
        void del (const T & value) {
            data.del(value);
        }
};
```



conjuntos

```
template <typename T>
class Set {
    private:
        openHashTable<T> data;
    public:
        unsigned size (void) {
            return data.size();
        }
        bool find (const T & value) {
            return data.find(value);
        }
        void add (const T & value) {
            data.add(value);
        }
        void del (const T & value) {
            data.del(value);
        }
};
```



conjuntos

```
template <typename T, typename C>
class Set : private C {
  public:
    using C::begin; using C::end; using C::size;
    using C::insert; using C::erase;
    bool find (const T & value) { return C::find(value) != end(); }
    void set_union (const Set & s) {
        for (auto value: s) insert(value);
    }
    Set set_intersection (const Set & s) {
        Set inter;
        for (auto value: s) if (find(value)) inter.insert(value);
        return inter;
    }
    Set set_difference (const Set & s) {
        Set diff;
        for (auto value: s) if (!find(value)) diff.insert(value);
        return diff;
    }
};
```

dicionários

```
template <typename K, typename V>
class DictElem : public pair<K, V> {
  public:
    DictElem (const K & key, const V & value) :
                    pair<K, V> (key, value) { }
    DictElem (const K & key) : pair<K, V> (key, V()) { }
    bool operator== (const DictElem<K, V> & other) const {
       return this->first == other->first;
    }
};

namespace std {
  template <typename K, typename V>
  struct hash<DictElem<K, V>> {
    size_t operator()(const pair<K,V> & element) const {
      return hash<K>()(element.first);
    }
  };
}
```

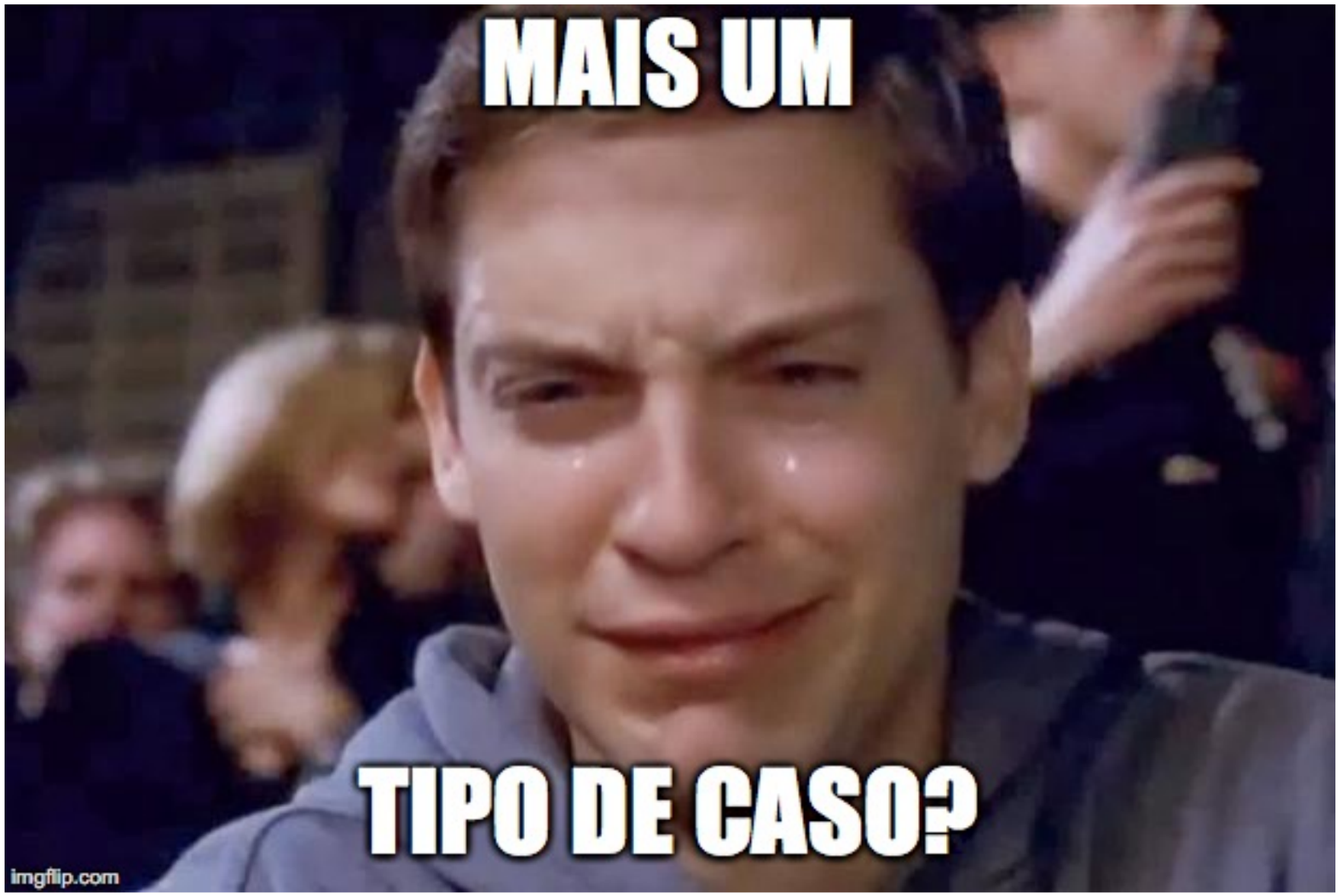
dicionários

```
template <typename K, typename V>
class Dict {
  private:
    typedef DictElem<K, V> T;
    unordered_set<T> data;
  public:
    void print (void) {
        for (auto element : data)
            cout << "(" << element.first << ", " << element.second << ") ";
        cout << endl;
    }
    void find (const K & key) {
        return data.find((T) key) != data.end();
    }
    void insert (const K & key, const V & value) {
        data.insert(T(key, value));
    }
    void erase (const K & key) {
        auto itr = data.find((T) key);
        if (itr != data.end()) data.erase(itr);
    }
```

dicionários

```
const V & operator[](const K & key) {
    auto itr = data.find((T) key);
    if (itr != data.end()) return itr->second;
    abort();
}
```

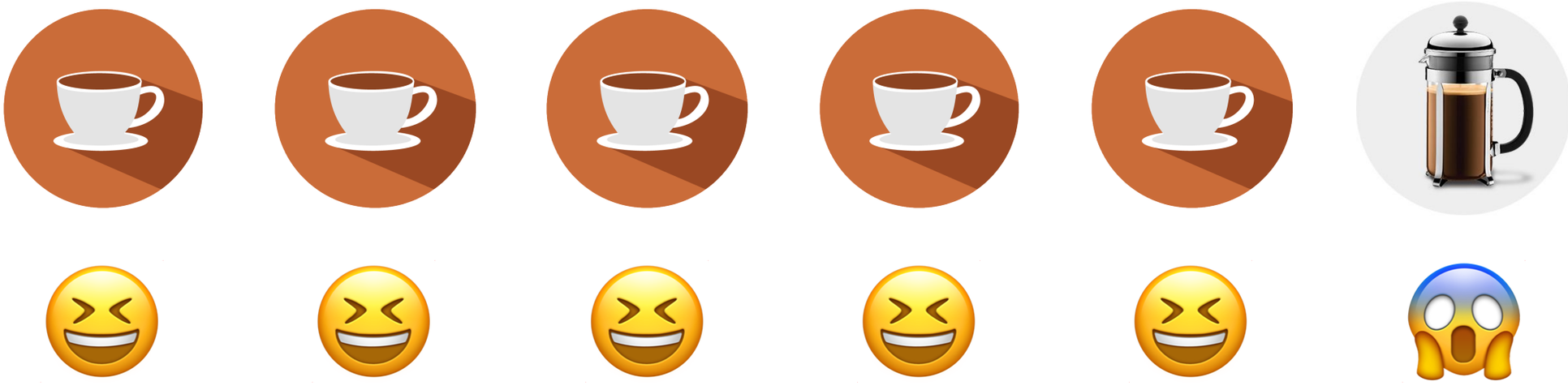
Operação	TAD Conjunto / Dicionário					
	Tabela de dispersão					
	Encadeamento simples			Endereçamento aberto		
	Melhor	Pior	Médio	Melhor	Pior	Médio

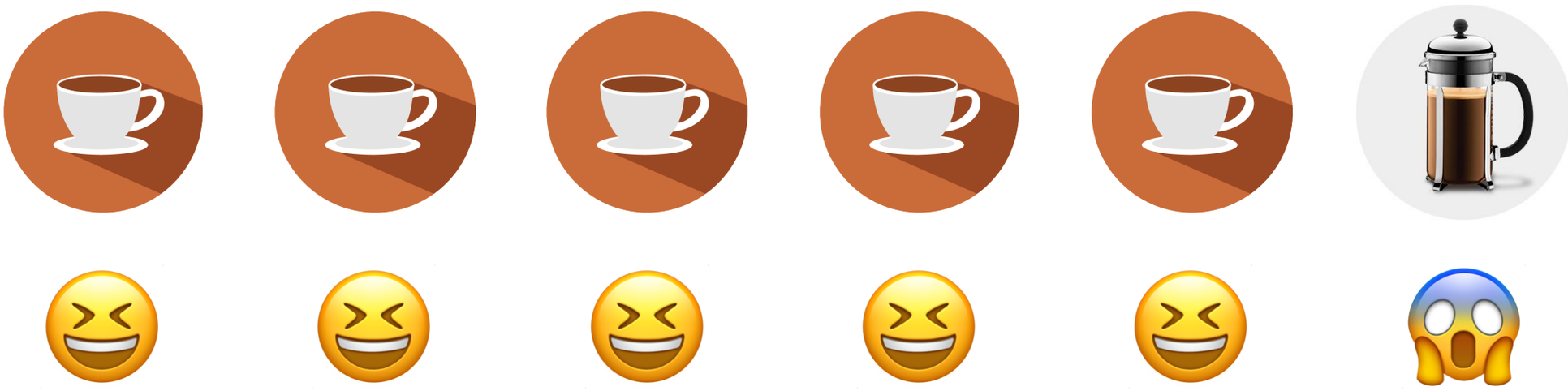

MAIS UM
TIPO DE CASO?
imgflip.com

caso **médio**

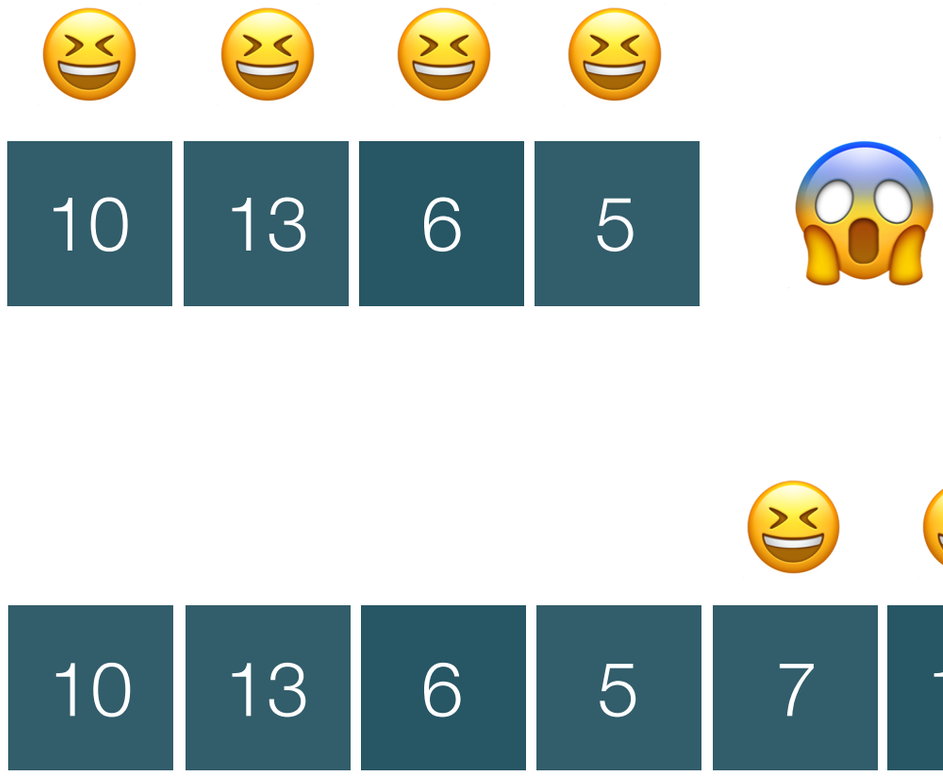

ADORO CAFÉ
MAS VOU TOMAR SÓ
UMA XÍCARA...


@abenzoados
o que aconteceu com ele?
acabou o café





complexidade amortizada



Operação	TAD Conjunto / Dicionário					
	Tabela de dispersão					
	Encadeamento simples			Endereçamento aberto		
	Melhor	Pior	Média	Melhor	Pior	Média
adicionar	Θ(1)	Θ(n)	Θ(1)	Θ(1)	Θ(n)	Θ(1)
remover	Θ(1)	Θ(n)	Θ(1)	Θ(1)	Θ(n)	Θ(1)
pertinência	Θ(1)	Θ(n)	Θ(1)	Θ(1)	Θ(n)	Θ(1)