T3: WebAssembly

Javiera Inostroza, Elías Sabja, Samuel Zúñiga

Dado un conjunto A, encontrar su conjunto potencia

Ejemplo: Conjunto potencia de {a,b,c}:

{ {}, {a}, {b}, {c}, {a,b}, {a,c}, {b,c}, {a,b,c} }

$$O(n * 2^n)$$

DEMO

```c
void printPowerSet(char *set, int set_size)
{
  /*set_size of power set of a set with set_size
    n is (2**n -1)*/
  unsigned int pow_set_size = pow(2, set_size);
  int counter, j;

  /*Run from counter 000..0 to 111..1*/
  for(counter = 0; counter < pow_set_size; counter++)
  {
    for(j = 0; j < set_size; j++)
     {
    /* Check if jth bit in the counter is set
       If set then print jth element from set */
    if(counter & (1<<j))
      printf("%c ", set[j]);
     }
     printf("\n");
  }
}
```

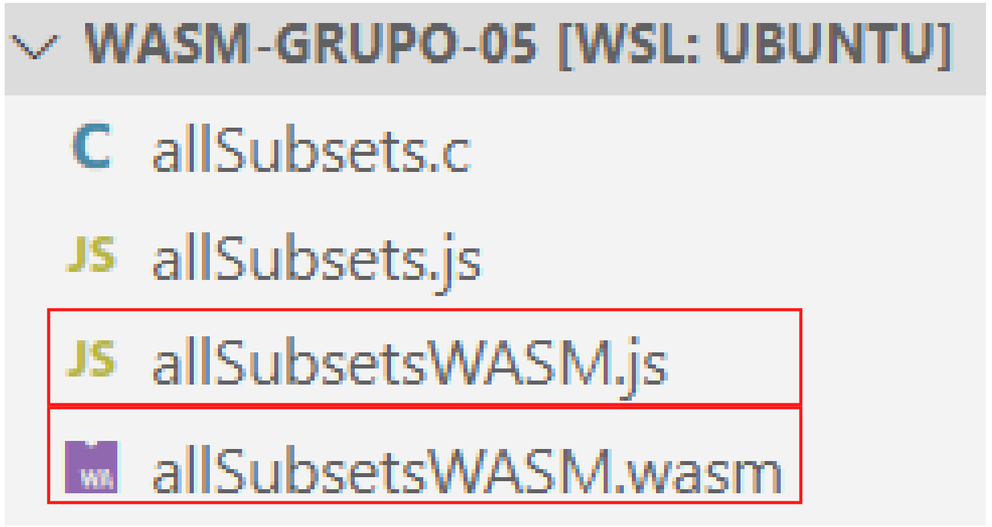

```javascript
const get_bit = (num, bit) => {
    let temp = (1 << bit);
    temp = temp & num;
    if (temp === 0) {
        return 0;
    }

    return 1;
};

export function get_all_subsets(v, sets) {
    let subsets_count = Math.pow(2, v.length);
    for (let i = 0; i < subsets_count; i++) {
        let st = new Set([]);
        for (let j = 0; j < v.length; j++) {
            if (get_bit(i, j) === 1) {
                st.add(v[j]);
            }
        }
        console.log(...st)

        sets.push(st);
    }
};
```

```
emcc -O3 allSubsets.c -o allSubsetsWASM.js -s
EXPORTED_FUNCTIONS="['_printPowerSet', '_malloc']" -s
EXPORTED_RUNTIME_METHODS=ccall,cwrap
```


WASM-GRUPO-05 [WSL: UBUNTU]
allSubsets.c
allSubsets.js
allSubsetsWASM.js
allSubsetsWASM.wasm

¿Cómo ocupar nuestra función de C en nuestro código en JS?

```
const cRunner = (arr, INPUT_LENGTH, Module) => {
        const ptrArray = new Int8Array(arr.length);
        const chars = arr.map(x => x.charCodeAt(0));
        for (let i = 0; i < arr.length; i++) { ptrArray[i] = chars[i]; }
        arrayBuffer = Module._malloc(arr.length * ptrArray.BYTES_PER_ELEMENT);
        Module.HEAP8.set(ptrArray, arrayBuffer);
        const secondsC = new Date().getTime();
        Module._printPowerSet(arrayBuffer, INPUT_LENGTH);
        const cTime = (new Date().getTime()) - secondsC;
        return cTime;
      };
```

```
const cRunner = (arr, INPUT_LENGTH, Module) => {
        const ptrArray = new Int8Array(arr.length);
        const chars = arr.map(x => x.charCodeAt(0));
        for (let i = 0; i < arr.length; i++) { ptrArray[i] = chars[i]; }
        arrayBuffer = Module._malloc(arr.length * ptrArray.BYTES_PER_ELEMENT);
        Module.HEAP8.set(ptrArray, arrayBuffer);
        const secondsC = new Date().getTime();
        Module._printPowerSet(arrayBuffer, INPUT_LENGTH);
        const cTime = (new Date().getTime()) - secondsC;
        return cTime;
      };
```

```
const cRunner = (arr, INPUT_LENGTH, Module) => {
        const ptrArray = new Int8Array(arr.length);
        const chars = arr.map(x => x.charCodeAt(0));
        for (let i = 0; i < arr.length; i++) { ptrArray[i] = chars[i]; }
        arrayBuffer = Module._malloc(arr.length * ptrArray.BYTES_PER_ELEMENT);
        Module.HEAP8.set(ptrArray, arrayBuffer);
        const secondsC = new Date().getTime();
        Module._printPowerSet(arrayBuffer, INPUT_LENGTH);
        const cTime = (new Date().getTime()) - secondsC;
        return cTime;
      };
```

```
1  const cRunner = (arr, INPUT_LENGTH, Module) => {
2          const ptrArray = new Int8Array(arr.length);
3          const chars = arr.map(x => x.charCodeAt(0));
4          for (let i = 0; i < arr.length; i++) { ptrArray[i] = chars[i]; }
5          arrayBuffer = Module._malloc(arr.length * ptrArray.BYTES_PER_ELEMENT);
6          Module.HEAP8.set(ptrArray, arrayBuffer);
7          const secondsC = new Date().getTime();
8          Module._printPowerSet(arrayBuffer, INPUT_LENGTH);
9          const cTime = (new Date().getTime()) - secondsC;
10         return cTime;
11       };
```

```
1  const cRunner = (arr, INPUT_LENGTH, Module) => {
2          const ptrArray = new Int8Array(arr.length);
3          const chars = arr.map(x => x.charCodeAt(0));
4          for (let i = 0; i < arr.length; i++) { ptrArray[i] = chars[i]; }
5          arrayBuffer = Module._malloc(arr.length * ptrArray.BYTES_PER_ELEMENT);
6          Module.HEAP8.set(ptrArray, arrayBuffer);
7          const secondsC = new Date().getTime();
8          Module._printPowerSet(arrayBuffer, INPUT_LENGTH);
9          const cTime = (new Date().getTime()) - secondsC;
10         return cTime;
11       };
```

```
1  const cRunner = (arr, INPUT_LENGTH, Module) => {
2          const ptrArray = new Int8Array(arr.length);
3          const chars = arr.map(x => x.charCodeAt(0));
4          for (let i = 0; i < arr.length; i++) { ptrArray[i] = chars[i]; }
5          arrayBuffer = Module._malloc(arr.length * ptrArray.BYTES_PER_ELEMENT);
6          Module.HEAP8.set(ptrArray, arrayBuffer);
7          const secondsC = new Date().getTime();
8          Module._printPowerSet(arrayBuffer, INPUT_LENGTH);
9          const cTime = (new Date().getTime()) - secondsC;
10         return cTime;
11       };
```

```
const cRunner = (arr, INPUT_LENGTH, Module) => {
        const ptrArray = new Int8Array(arr.length);
        const chars = arr.map(x => x.charCodeAt(0));
        for (let i = 0; i < arr.length; i++) { ptrArray[i] = chars[i]; }
        arrayBuffer = Module._malloc(arr.length * ptrArray.BYTES_PER_ELEMENT);
        Module.HEAP8.set(ptrArray, arrayBuffer);
        const secondsC = new Date().getTime();
        Module._printPowerSet(arrayBuffer, INPUT_LENGTH);
        const cTime = (new Date().getTime()) - secondsC;
        return cTime;
      };
```

```
const cRunner = (arr, INPUT_LENGTH, Module) => {
        const ptrArray = new Int8Array(arr.length);
        const chars = arr.map(x => x.charCodeAt(0));
        for (let i = 0; i < arr.length; i++) { ptrArray[i] = chars[i]; }
        arrayBuffer = Module._malloc(arr.length * ptrArray.BYTES_PER_ELEMENT);
        Module.HEAP8.set(ptrArray, arrayBuffer);
        const secondsC = new Date().getTime();
        Module._printPowerSet(arrayBuffer, INPUT_LENGTH);
        const cTime = (new Date().getTime()) - secondsC;
        return cTime;
      };
```

```
const jsRunner = arr => {
        const secondsJS = new Date().getTime();
        get_all_subsets(arr, []);
        const javascripTime = (new Date().getTime()) - secondsJS;
        return javascripTime;
      };
```

DIFICULTADES Y CONCLUSIONES

Difícil conectar (inicialmente) el código WASM con Javascript

Problemas con RequireJS:

MODULE NAME HAS NOT BEEN LOADED YET

Falta de documentación y claridad

WASM no siempre es más rápido que JS

Trade-off entre tiempo de carga de módulos, y tiempo de ejecución del algoritmo.

Sirve para utilizar código ya existente en el navegador.

T3: WebAssembly

Javiera Inostroza, Elías Sabja, Samuel Zúñiga