1 2SAT

```
function reduce(j,C) {
    C' ← C
    for c ∈ C' {
        if j ∈ c {
            remove c from C'
        }
        if −j ∈ c {
            remove −j from c
        }
    }
    return C'
}
```

As in class, represent clauses as lists of signed integers. To set $x[j] = 1$ ($x[j] = 0$, call reduce(j,C) (reduce(−j,C)).

Removing a literal from a clause is $O(1)$ because the clauses are constant size. Removing a clause from the clause set in constant time requires e.g. a linked list representation of the clause set. Total cost is $O(m)$.

2 Subset Sum

Let's represent our subproblems as the subset $U \subseteq S$ already put in the subset, the set $W \subset S \setminus U$ of remaining numbers, along with the total t we are trying to achieve.

```
function expand(W,U,t) {
    Let w ∈ W.
    Let W' ← W \ {w}
    return {(W', U ∪ {w}, t), (W', U, t)}
}
```

Obviously we keep the same representation for test.

```
function test(W,U,t) {
    Let σ_W = ∑_{w∈W} w
    Let σ_U = ∑_{u∈U} u

    if σ_U > t return FAIL                    # this also catches negative t
    if σ_U = t return SUCCESS

    # optional test
    if t − σ_u > σ_w return FAIL
    return UNKNOWN
}
```

3 Independent Set

We can use a somewhat similar representation: U is the set of vertices definitely in the independent set, S is the set of remaining candidates, and k the size of independent set we are trying to achieve. Let $G = (V, E)$ be a global variable, with $n = |V|$, $m = |E|$. We fix that $P_0 = (V, \emptyset, k)$.

3.1 Expand

```
function expand(S, U, k){
  Let v ∈ S                                   # O(1)
  S' ← S \ {v}                                # O(log n)
  for (v, w) ∈ E {
    S' ← S' \ {w}
  }                                           # O(m log n) loop

  return {(S', U ∪ {v}, k), (S \ {v}, U, k)}  # O(log n)
}
```

3.2 Test

Total complexity is $O(m \log n)$.

```
function test(S, U, k) {
  if |U| = k return SUCCESS
  if S = ∅ return FAIL # out of elements to add
  return UNKNOWN
}
```

For reasonable set representation, this is $O(1)$.

3.3 Proof

This is a decision problem, so we just need to make sure that the algorithm correctly reports YES and NO answers.

3.3.1 YES case

For the YES case, let's prove by induction that U is always an independent set.

Base case The empty set is an independent set.

Induction Suppose that for all $|U| < n$ returned by extend are independent sets. Now consider calling extend on a set of size $n - 1$. We either keep U unchanged, in which case the answer is clear, or we add v to U. We know that v is not adjacent to any previously added vertex, since the **for** loop would have removed it from S already. So the U returned is indeed an independent set.

3.3.2 NO case

Suppose the algorithm returns FAIL, but there really is some independent U^* set of size k. Let U' be the largest subset of U^* for which test returns FAIL. We know in that case $|U'| < k$, but $S = \emptyset$. That means every element of $V \setminus U'$ is adjacent to some element of U', which contradicts the existence of U^*.