Ex 1. 1. For 2^{64} operations, it needs about $\frac{2^{64}}{2^{50}} = 2^{14}$s, and for 2^{80} operations, 2^{30}s is needed

2. For 2^{64} operation, it needs $\frac{2^{64}}{24*3600*3.8*2^{30}} \approx 52327$ desktops. And for 2^{80} operations for a month, it needs, about 114309202 desktops.

3. For 2^{64} bits, it needs $\frac{2^{64}}{2^44} = 2^{20}$ hardrives, and for 2^{80} it needs about 2^{36} hardrives.

Ex 2. It can be done in the following way

Let us use an array R to store the subset. For the first k subsets, directly put them into R. For the items afterwards, suppose for number m element $(k < m < n)$, generate a random number i ranging from 1 to m. If the number is in the range $[1, k]$, then replace $R[i]$ by the m_{th} element, otherwise do nothing. When all items are checked, the subset is generated.

Ex 3. 1. Pseudo code can be shown below

Algorithm 1 Triangle Layer Sum

Input: Number of Layer i
Output: The sum of elements on the i_{th} layer

```
function ThreePower(i)
    if i = 1 then
        return 1
    end ifreturn 3 × ThreePower(i − 1)
end function
return ThreePower(i)
```

2. It is a $O(n)$ algorithm and for its correctness.

According to the rule, we can see that to generate the elements in layer $i + 1$, the elements in layer i are used 3 times except the two 1s in both sides — They are used 2 times. And because there are two new 1s in the layer, the recurrance relation can be written as

$$f_n = 3(f_{n-1} - 2) + 2 \times 2 + 2 = 3f_{n-1}$$

And because $f_1 = 1$, then it can be seen that $f_n = 3^{n-1}$

Ex 5. 2. Given a subset of points, it uses at worst $O(n^2)$ time to check whether the points are adjacent to each other by brute force, so an answer can be checked in polynomial time, then it is an $\mathcal{NP}$ problem

3. Let G be a graph whose vertices are (v, c), where c is one of clauses and v is a literal appears in that clause (with or without negation), and denote (v_1, c_1), (v_2, c_2) as connected if $c_1 \neq c_2$ and u is not $\neg v$. Then the existance of k-clique in the graph can be determined by whether $\mathcal{F}$ is satisfied

4. The problem is $\mathcal{NP}$-complete

Ex 6. 2. It is the problem to determine whether a undirected graph has a independent subset of size k

3. Given a subset of points, it uses at worst $O(n^2)$ time to check whether the points are not adjacent to each other by brute force, so an answer can be checked in polynomial time, then it is an $\mathcal{NP}$ problem

4. Just using the complementary graph of G can reduce the problem to clique problem
5. It is also $\mathcal{NP}$-complete