Ex 1. 1. (a) First is to prove that there exists a number x_0 such that for $x > x_0$, $x^3 - 3x^2 - x + 1 \geq cx^3$, which means that $\frac{x^3}{x^3-3x^2-x+1} \leq \frac{1}{c}$.

Because $\lim\limits_{x\to\infty} \frac{x^3}{x^3-3x^2-x+1} = 1$, which means that for any number $\delta > 0$, there exists a number ϵ that $\frac{x^3}{x^3-3x^2-x+1} < 1 + \delta$ when $x > \epsilon$

Let $c = \frac{1}{1+\delta}, x_0 = \epsilon$, it can be easily seen that $n^3 - 3n^2 - n + 1 = \Omega(n^3)$

And $n^3-3n^2-n+1 = O(n^3)$ can be proved in the similar way. So it can be concluded that $n^3-3n^2-n+1 = \Theta(n^3)$

(b) It is to prove that there exists a number x_0 such that for $x > x_0$, $x^2 \leq c2^x$, which is to say that $\frac{x^2}{2^x} < c$.

Extract the logarithm, the inequality will be $2lnx - xln2 < lnc$, which is $\frac{2lnx-lnc}{x} < ln2$

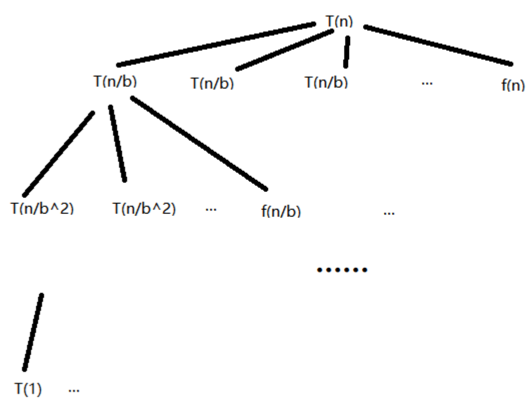
Because $\lim\limits_{x\to\infty} \frac{2lnx-lnc}{x} = 0$, it is obvious that such x_0 exists, so $n^2 = O(2^n)$

2. (a) $n\sqrt{n} = O(n^2 - 1)$

3. (a) There is no such pair. (b) $f(n) = n^2 - 1, g(n) = n\sqrt{n}$

4. f_2, f_3, f_1, f_4

Ex 2. 1. (a) The recursion tree is shown below.



(b) i. The depth is $log_b n + 1$

ii. Number of leaves is $\sum_{k=0}^{log_b n} a^k = \frac{a^{log_b n+1}-1}{a-1}$

iii. Suppose the depth of the root is 0, then the cost on the k_{th} depth is

$$a^{k-1} f(\frac{n}{b^{k-1}})$$

for $0 < k < log_b n$, 0 for $k = 0$ and $a^{log_b n-1} f(b) + a^{log_b n} T(1)$ for $k = log_b n$

iv. It can be seen that

$$T(n) = \sum_{k=1}^{log_b n} a^{k-1} f(\frac{n}{b^{k-1}}) + a^{log_b n} T(1) = \frac{T(1)}{a} n^{log_b a} + \sum_{j=0}^{log_b n-1} a^j f(\frac{n}{b^j}) = \Theta(n^{log_b a}) + \sum_{j=0}^{log_b n-1} a^j f(\frac{n}{b^j})$$

2. (a). i. It can be seen that $c_1(\frac{n}{b^j})^{log_b a} \leq f(\frac{n}{b^j}) \leq c_2(\frac{n}{b^j})^{log_b a}$ for $n > n_0$

So $c_1 \sum_{j=0}^{log_b a-1} a^j(\frac{n}{b^j}) \leq g(n) \leq c_2 \sum_{j=0}^{log_b a-1} a^j(\frac{n}{b^j})$ when $n > n_0$, and consequently $g(n) = \Theta(\sum_{j=0}^{log_b a-1} a^j(\frac{n}{b^j})^{log_b a})$

iii. Because actually

$$\sum_{j=0}^{log_b a-1} a^j (\frac{n}{b^j})^{log_b a} = n^{log_b a} log_b n$$

So

$$g(n) = \Theta(n^{log_b a} log_b n)$$

(b). ii. This is the partial sum of a geometric sequence starting from $n^{log_b a-\epsilon}$, with ratio $\frac{a}{b^{(log_b a-\epsilon)}}$ and $log_b n$ terms so it is

$$n^{log_b a-\epsilon} \frac{\frac{a}{b^{(log_b a-\epsilon)}}^{log_b n} - 1}{\frac{a}{b^{(log_b a-\epsilon)}} - 1} = \frac{n^\epsilon - 1}{b^\epsilon - 1} n^{log_b a-\epsilon}$$

iii. From 1 and 2, it can be concluded that $g(n) = O(\frac{n^\epsilon - 1}{b^\epsilon - 1} n^{log_b a-\epsilon}) = O(n^{log_b a})$

(c).i. It is obvious because $g(n)$ has a $f(n)$ term in it, and all terms are larger than or equal to 0, so it is always larger than or equal to $f(n)$

ii. $a^j f(\frac{n}{b^j}) \leq a^{j-1} c f(\frac{n}{b^{j-1}}) \leq ... \leq c^j f(n)$

iv. Because $g(n)$ is both $O(f(n))$ and $\Omega(f(n))$, so it is $\Theta(f(n))$

Ex 3. The algorithm is shown below.

Algorithm 1 Ramanujam Number Finding

Input: A number n
Output: All Ramanujam numbers smaller than n

$T \leftarrow \{\}$
for $i = 1 \to n$ **do**
  for $j = 1 \to \lfloor \sqrt[3]{i} \rfloor$ **do**
    if $i - j^3$ is a cubic number **then**
      for $k = j + 1 \to \sqrt[3]{i - j^3} - 1$ **do**
        if $i - k^3$ is a cubic number **then**
          $T \leftarrow T + \{i\}$
        end if
      end for
    end if
  end for
end for
return T

The complexity is $O(n(\sqrt[3]{n})^2)$

Ex 4. Suppose only pirate 5 and 6 remains. Of course pirate 6 will not agree with pirate 5 until pirate 5 give all coins to pirate 6, so the distribution must be zero for pirate 5 and 300 for pirate 6.

Then consider the situation when there are pirate 4, 5 and 6. Winning the vote from pirate 5 is very easy for pirate 4 because he only need to give pirate 5 one coin, which is larger than his profit if pirate 4 is dead. And he need not to care pirate 6 because he has won pirate 5's vote. So the distribution will be 299 for pirate 4, 1 for pirate 5 and 0 for pirate 6.

When there are pirate 3,4,5,6, what pirate 3 should do is to raise profit of pirate 6 to 1 and then give pirate 4 and 5 no coin. So the distribution will be 299 for pirate 3, 0 for pirate 4 and 5, 1 for pirate 6.

When there are pirate 2,3,4,5,6, what pirate 2 should do is to raise profit of pirate 4 and 5 to 1 and then give pirate 3 and 6 no coin. So the distribution will be 298 for pirate 2, 0 for pirate 3 and 6, 1 for pirate 4 and 5.

When there are pirate 1,2,3,4,5,6, what pirate 1 should do is to raise profit of pirate 3 and 6 to 1 and then give pirate 2,4 and 5 no coin. So the final distribution will be 298 for pirate 1, 0 for pirate 2,4 and 5, 1 for pirate 3 and 6. Pirate 1 will win the vote from pirate 3,6 and himself