গণিত বা কম্পিউটার বিজ্ঞানের সব সমস্যাই কি সমাধানযোগ্য? আমরা জানি NP ক্যাটাগরির সমস্যাগুলোকে পলিনোমিয়াল সময়ে সমাধান করার সম্ভব নাকি সেটা জানা এখন পর্যন্ত সম্ভব হয় নি, কিন্তু ইনপুটের আকার যথেষ্ট ছোট হলে অথবা তোমার হাতে অসীম সময় এবং মেমরি থাকলে NP সমস্যাও এক্সপোনেনশিয়াল সময়ে সমাধান করা সম্ভব। কিন্তু এমন কিছু সমস্যা আছে যেটা তোমার হাতে যত বড় সুপার কম্পিউটারই থাকুক সমাধান করা সম্ভব না। এখানে আমি ধরে নিচ্ছি আমাদের কম্পিউটারগুলো টুরিং মেশিন কম্পিউটেবল। (টুরিং মেশিন কি মনে না থাকলে আগে আমার এই লেখাটা পড়ো)
হাল্টিং প্রবলেম (Halting Problem) এমনই একটা সমস্যা যার কোনো সমাধান নেই। তোমাকে একটা কম্পিউটার প্রোগ্রাম এবং একটা ইনপুট দেয়া হলো। তোমাকে বলতে হবে সেই ইনপুটের জন্য প্রোগ্রামটা কি সীমিত সময়ে থামবে নাকি অসীম বা ইনফিনিটি লুপে আটকে যাবে?

```
def my_program(x):
    count = 0
    while True:
        if count == x:
            break
        count = count + 1
```

উপরের কোডে তুমি যদি $x=-10x=-10$ ইনপুট দাও তাহলে প্রোগ্রামটা কখনোই থামবে না (কোডটা পাইথনে লেখা, ইন্টিজার ওভারফ্লো এর ভয় নেই)
তোমার কাজ হলো এমন একটা প্রোগ্রাম লেখা যেটা যেকোনো আরেকটা প্রোগ্রাম এবং ইনপুট দেখে বলে দিতে পারবে প্রোগ্রামটা ইনফিনিটি লুপে আটকে যাবে নাকি। এটাই হলো হাল্টিং প্রবলেম (Halting Problem)।

এখন আমরা "প্রুফ বাই কনট্রাডিকশন" মেথডের সাহায্যে প্রমাণ করবো যে হাল্টিং প্রবলেম সমস্যার সমাধান করা সম্ভব না। আমরা প্রথমে ধরে নিবো যে সমস্যাটির একটা সমাধান আছে এবং তারপর প্রমাণ করবো যে সমাধানটি অস্তিত্ব থাকা অসম্ভব।

ধরো আমাদের কাছে একটা প্রোগ্রাম আছে যেটা হাল্টিং প্রবলেম সমাধান করতে পারে। প্রোগ্রামটা হতে পারে এরকম:

```
will_it_halt(P, I)
  if P(I) halts in finite time
     return TRUE
  else
     return FALSE

```

এটা একটা কাল্পনিক কোড। কোডটার প্যারামিটার হলো একটা প্রোগ্রাম P এবং P কে প্যারামিটার হিসাবে পাঠানো হবে এমন একটা ইনপুট I। কোডটা কোন উপায়ে বলে দিতে পারে P প্রোগ্রামে I ইনপুট পাঠানো হলে সেটা সীমিত সময়ে থামবে নাকি থামবে না।

এ ধরণের কোডকে বলা হয় Oracle Machine। Oracle এর আক্ষরিক অর্থ হলো একজন ওঝা যে আধ্যাত্মিক ক্ষমতা দিয়ে সমস্যা সমাধান করতে পারে। আমাদের will_it_halt কোডও জাদুকরি ক্ষমতা দিয়ে হাল্টিং প্রবলেম সমাধান করতে পারে। এখন আমরা যদি প্রমাণ করতে পারি এমন কোনো Oracle Machine থাকা সম্ভব না যেটা দিয়ে হাল্টিং প্রবলেম সমাধান করা যায় তাহলেই আমাদের কাজ শেষ।
এখন আমরা আরেকটা প্রোগ্রাম লিখবো, মনে করো প্রোগ্রামটার নাম প্যারাডক্স (Paradox)।

```
1 paradox(program)
2     will_it_halt(program, program)
3        Run Forever
4     else
5        return TRUE
6
```

প্যারাডক্স ইনপুট হিসাবে একটা প্রোগ্রামকে গ্রহণ করে। একটা প্রোগ্রাম হলো কিছু স্ট্রিং দিয়ে লেখা কিছু ইনস্ট্রাকশন। তাই আমরা চাইলে প্যারামিটার হিসাবে প্যারাডক্স প্রোগ্রামটাকেই পাঠাতে পারি!

```
1 paradox(paradox)
```

এখন দ্বিতীয় লাইনে আমরা ওরাকল মেশিনকে জিজ্ঞেস করছি প্যারাডক্স প্রোগ্রামে ইনপুট হিসাবে প্যারাডক্স প্রোগ্রামটাকেই পাঠালে সেটা ইনফিনিটি লুপে আটকে যাবে নাকি যাবে না। যদি ওরাকল বলে যে ইনফিনিটি লুপে আটকে যাবে না, তাহলে আমরা ৩য় লাইনে সেটাকে ইনফিনিটি লুপে আটকে দিবো! আর ওরাকল যদি বলে যে ইনফিনিটি লুপে আটকে যাবে তাহলে আমরা প্রোগ্রাম থেকে বের হয়ে যাবো। তারমানে ওরাকল যা বলছে সেটা সত্যি না, বরং তার উল্টাটা ঘটছে। এটাই হলো কন্ট্রাডিকশন, এ থেকে আমরা বলতে পারি will_it_halt নামক ওরাকল মেশিনটার অস্তিত্ব থাকা সম্ভব না!

এটাই হলো হাল্টিং প্রবলেম যা সমাধানযোগ্য না সেটার প্রমাণ।

মোবাইলে কিছু কিছু অ্যাপ চালু করলে কোনো একটা বাগের জন্য স্ক্রীন ফ্রিজ হয়ে যায়, তখন ফোনের ব্যাটারি খুলে রিস্টার্ট করা ছাড়া উপায় থাকে না। এমন কোনো অ্যাপ কি তুমি লিখতে পারবে যেটা বলে দিতে পারবে কোন অ্যাপ চালালে স্ক্রীণ ফ্রিজ হয়ে যেতে পারে? এটা হাল্টিং প্রবলেমেরই একটা ভ্যারিয়েশন যা এই লেখাটায় সুন্দর করে ব্যাখ্যা করা আছে, আগ্রহ থাকলে পড়তে পারো।

হাল্টিং প্রবলেম যদি সমাধানযোগ হতো তাহলে কি হত? তাহলে কোড ডিবাগিং করা অনেক সহজ হয়ে যেতে, কোড চালু না করেই আমরা বলতে পারতাম কোডটা ইনফিনিটি লুপে পড়বে নাকি। এর থেকেও গুরুত্বপূর্ণ ব্যাপার হলো, অনেক গাণিতিক সমস্যা সমাধান করা যেত এই will_it_halt প্রোগ্রামটা দিয়ে। একটা উদাহরণ দেই। গোল্ডবাখ নামক একজন গণিতবিদ বহু বছর আগে বলে গেছেন “২ এর থেকে বড় যে কোন জোড় সংখ্যাকে দুটি প্রাইম সংখ্যার যোগফল হিসাবে লেখা সম্ভব”। কেও এখনো প্রমাণ করতে পারে নি যে কনজেকচারটা সত্য নাকি মিথ্যা। এখন আমি একটা প্রোগ্রাম লিখলাম প্রমাণ করার জন্য:

```
def goldbach():
  k = 4
  while True:
    ok = False
    for p1  in range(k):
     for p2 in range(k):
      if(prime(p1) && prime(p2) && p1+p2==k):
        ok = True
    if not ok:
      exit()
    k += 2
```

এই প্রোগ্রামটা k=4 থেকে শুরু করে প্রতিটি জোড় সংখ্যাকে ব্রুট ফোর্সের মাধ্যমে দুটি প্রাইম সংখ্যার যোগফল হিসাবে লেখার চেষ্টা করবো। যদি কোনো k এর জন্য ok = false হয় তাহলে প্রমাণ হবে যে কনজেকচারটা মিথ্যা, তখন প্রোগ্রামটা বন্ধ হয়ে যাবে। আর যদি কনজেকচারটা সত্যি হয় তাহলে অসীম সময় ধরে প্রোগ্রামটা চলতে থাকবে। will_it_halt প্রোগ্রামটার অস্তিত্ব থাকলে এখন will_it_halt(goldback, null) এর আউটপুট দেখেই বলে দিতে পারতাম কনজেকচারটি সত্য নাকি মিথ্যা!

তুমি যদি এমন একটা গাণিতিক মডেল বের করতে পারো যা হাল্টিং প্রবলেমকে সমাধান করতে পারে তাহলে আমরা আমরা একটা হাইপার কম্পিউটেশন বা **super-Turing computation** মডেল পাবো, তখন আমরা এমন সব সমস্যা সমাধান করতে পারবো যা টুরিং মেশিন দিয়ে সমাধান করা সম্ভব না।

হাল্টিং প্রবলেম অনেক দার্শনিক প্রশ্নেরও জন্ম দিয়েছে। যেমন আমরা কি কখনো নিজের মস্তিষ্ক সম্পর্কে ১০০% জানতে পারবো? নাকি কোনো থিওরিটিকাল সীমাবদ্ধতার কারণে মস্তিষ্কের অনেক রহস্য আমরা কোনো জানতে পারবো না?

আজ এখানেই শেষ, হ্যাপি কোডিং!

রেফারেন্স

http://www.cgl.uwaterloo.ca/csk/halt/
https://www.quora.com/If-the-halting-problem-was-solvable-what-would-be-the-implications