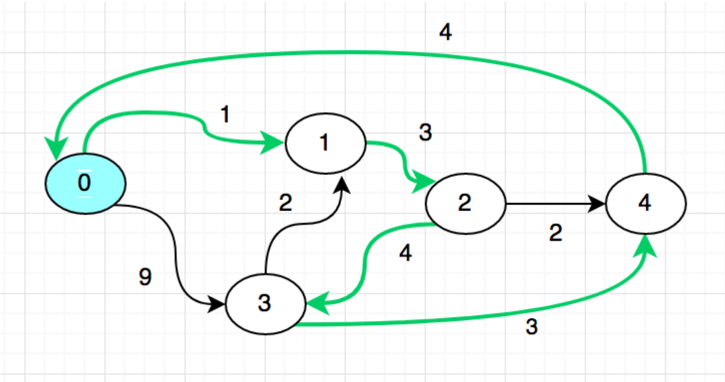
(সবগুলো পর্ব) বিটমাস্ক ব্যবহার করে বেশ কিছু NP-Complete/NP-Hard প্রবলেম সলভ করা যায়। বিটমাস্কের ব্যবহার জানা থাকলে এটা তেমন কঠিন কিছু নয়। এই লেখা পড়ার আগে তোমার জানা লাগবে কিভাবে বিট ম্যানিপুলেশন করতে হয়, সেজন্য তুমি এই লেখাটা পড়তে পারো। এই প্রবলেম সলভ করতে গ্রাফ থিওরি জানার দরকার নেই।

আজকে আমরা বিখ্যাত ট্রাভেলিং সেলসম্যান প্রবলেম বিটমাস্ক দিয়ে সলভ করবো। প্রবলেমটা হলো তোমাকে কিছু শহর এবং রাস্তা একটা গ্রাফ হিসাবে দেয়া আছে। তোমাকে প্রথম শহর থেকে শুরু করে সবগুলো শহর ঠিক একবার করে ভ্রমণ করে প্রথম শহরে ফিরে আসতে হবে। প্রশ্ন হলো সর্বনিম্ন কত দূরত্ব অতিক্রম করে তুমি কাজটা করতে পারবে?



ছবির গ্রাফে n=4n=4 টি শহর আছে এবং অপটিমাল পথ হলো 0–>1–>2–>3–>4–>00–>1–>2–>3–>4–>0, মোট দূরত্ব 1515।

ট্রাভেলিং সেলসম্যান NP এবং NP-Hard গ্রুপের প্রবলেম, অর্থাৎ প্রবলেমটি NP-complete। এর মানে এই প্রবলেমের কোনো পলিনোমিয়াল সলিউশন আমাদের জানা নেই। সব সলিউশনই এক্সপোনেনশিয়াল কমপ্লেক্সিটির।

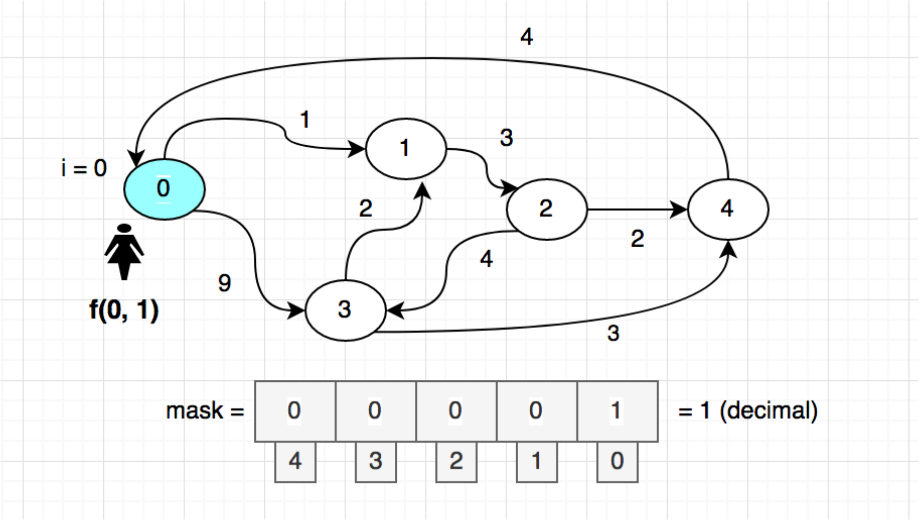
প্রথমেই আমরা অন্যান্য ডিপি প্রবলেমের মত সাবপ্রবলেম খোজা শুরু করি।

আমরা f(i)f(i) একটা ফাংশন ডিফাইন করতে পারি যেটা হলো ii তম শহর থেকে শুরু করে মাঝখানের সব শহর ভ্রমণ করে সর্বশেষ শহরে পৌছানোর দূরত্ব। কিন্তু সমস্যা হলো এক শহর দুইবার ভ্রমণ করা যাবে না, তাই আমাদের জানা দরকার কোন কোন শহর আমরা ইতোমধ্যেই ভ্রমণ করেছি। এখানেই বিটমাস্ক আমাদের সাহায্য করবে।

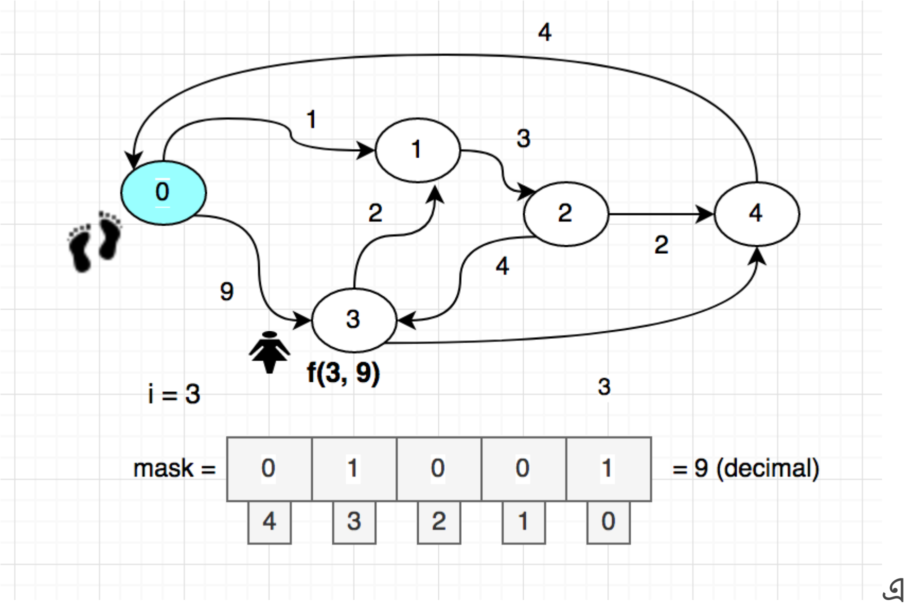
আমরা জানি একটা ইন্টিজারে ৩২টা বিট থাকে। আমরা এই বিটগুলোকেই ব্যবহার করবো কোন শহরে গেছি আর কোন শহরে যাইনি সেটা বুঝাতে। যদি ii তম শহরে যাই তাহলে ii হত বিট টা অন করে দিবো, অর্থাৎ 11 বানিয়ে দিবো। এই ইন্টিজারটাকেই বলা হয় বিটমাস্ক।

এবার তাহলে আমরা ফাংশনে আরেকটা প্যারামিটার অ্যাড করতে পারি: f(i,mask)f(i,mask)। ii দিয়ে বুঝাচ্ছে বর্তমানে তুমি কোন শহরে আছো, maskmask এর ভিতর থাকবে তুমি কোন কোন শহর ভ্রমণ করেছো সেটার তথ্য।

উপরের উদাহরণে 55 টা শহর আছে, তাই বিট লাগবে 55 টা। আমরা ১টা ইন্টিজারের শেষ 55 টা বিট ব্যবহার করবো। শুরুতে আমরা শহর 00 তে আছি এবং শুধুমাত্র ওই শহরটাই ভ্রমণ করেছি। তাহলে মাস্ক হবে বাইনারি 0000100001, অর্থাৎ খালি 00 তম বিটটা অন থাকবে।



এখন যদি তুমি 00 থেকে 33 এ যাও তাহলে কি হবে?



এবার আমরা 33 নম্বর বিটটাকেও অন করে দিয়েছি। বাইনারি 0100101001 কে ডেসিমালে 99 লেখা যায়, তাই বর্তমান স্টেট হবে f(3,9)f(3,9)।

কি ঘটছে বুঝে গেছো এতক্ষণে। আমরা ii তম শহর থেকে অ্যাডজেসেন্ট যেসব শহরে আগে ভ্রমণ করিনি সেসব শহরে যাবো এবং যাওয়ার সময় ওই শহরের বিটটি অন করে দিবো। যেদিক দিয়ে গেলে মিনিমাম রেজাল্ট পাবো সেটাই আমাদের উত্তর।

ii তম বিট অন করার উপায় কি? সেটা বিস্তারিত বলা আছে আর্টিকেলের প্রথমেই যে লিংক দিয়েছি সেখানে। তারপরেও আরেকবার মনে করিয়ে দেই। একটা সংখ্যা xx এর ii তম বিট অন করার নিয়ম হলো এমন একটা সংখ্যার সাথে OR অপারেশন চালানো যার ii তম বিটে 11 আছে কিন্তু বাকি সব বিট শুন্য।

```
int turnOn(int x,int pos) {
    return x | (1<<pos);
}
```

যদি ii বিট টা 00 নাকি 11 চেক করতে চাও তাহলে এমন একটা সংখ্যার সাথে AND করতে হবে যার ii তম বিটে 11, বাকি সব বিট শুন্য। যদি রেজাল্ট পজিটিভ হয় তাহলে ওই বিট অন আছে।

```
bool isOn(int N,int pos) {
    return (bool)(N & (1<<pos));
}
```

আসলে প্রবলেমে ফিরে আসি। আমাদেরকে f(0,1)f(0,1) প্রবলেমটা সলভ করতে হবে। রিকার্সন শেষ হবে যখন f(0,2^{n-1})f(0,2n−1) এ পৌছাবো। 2^{n-1}2n−1 কোথা থেকে আসলো? nn টা বিট অন থাকা মানে সবগুলো শহর ঘোরা শেষ, আর যে সংখ্যার প্রথম nn টা বিট অন, বাকিগুলো অফ সেই সংখ্যাটাই 2^{n-1}2n−1। যেমন 2^3–1=723–1=7 এর বাইনারি 111111।

রিকার্সিভ ফর্মূলাটা

$$f(i, 2^{n-1}) = dis[i][0]$$

$$f(i, mask) = min(f(j, turnOn(mask, j) + w(i, j))\ where(i, j) \in E$$

এখানে w(i,j)w(i,j) দিয়ে শহর দুটির দূরত্ব বুঝিয়েছে।

সি++ কোড

কোড লেখার সময় ধরে নিচ্ছি ww হলো একটা n×nn×n অ্যারে, w[i][j]w[i][j] তে ii এবং jj এর দুরত্ব রাখা আছে, যদি সরাসরি যাবার পথ না থাকলে তাহলে infinity বসিয়ে দিতে পারো।

```
#define EMPTY_VALUE -1
#define MAX_N 10
#define INF 1061109567

int w[MAX_N][MAX_N];
int mem[MAX_N][1<<MAX_N];

int turnOn(int x, int pos) {
    return N | (1<<pos);
}

bool isOn(int x ,int pos) {
    return (bool)(x & (1<<pos));
```

```
}

int n;
int f(int i, int mask) {
    if (mask == (1<<n) - 1) {
        return w[i][0];
    }

    if (mem[i][mask] != -1) {
        return mem[i][mask];
    }

    int ans = INF;
    for (int j = 0;j < n;j++) {
        if (w[i][j] == INF) continue;

        if (isOn(mask,j) == 0) {
            int result = f(j, turnOn(mask, j)) + w[i][j];
            ans = min(ans, result);
        }
    }

    return mem[i][mask] = ans;
}
```

কমপ্লেক্সিটি

আমাদের n টা বিট থাকলে সেটা দিয়ে কয়টা সংখ্যা রিপ্রেজেন্ট করা যায়? প্রতি বিটের জন্য দুইটা করে অপশন, মোট পসিবিলিটি 2^n। তারমানে $mask$ এর ভ্যালু হতে পারে 0 থেকে 2^n পর্যন্ত। তারমানে স্টেট আছে $n \times 2^n$ টা। ভিতরের লুপটার জন্য মোট কমপ্লেক্সিটি হবে $O(n^2 \times 2^n)$। খুব একটা ভালো কমপ্লেক্সিটি না, তবে np-complete প্রবলেম হিসাবে একদম খারাপ না (আরো ভালো অ্যালগরিদম চাইলে Branch and Bound নিয়ে গুগল করো)।

কোন প্রবলেমে যখনই দেখবে n বেশ ছোট এবং তোমার সবগুলো আইটেমের স্টেট সেভ করা লাগছে, তখনই বিটমাস্ক দিয়ে সলভ করা যায় নাকি চিন্তা করে দেখবে। সবসময় যে করা যাবে সেটা না, তবে এখন তোমার হাতে নতুন একটা প্রবলেম সলভিং টুলস আছে, চেষ্টা করে দেখতে দোষ কি?

প্র্যাকটিস প্রবলেম

https://leetcode.com/problems/partition-to-k-equal-sum-subsets/
https://leetcode.com/problems/find-the-shortest-superstring/
acm.timus.ru/problem.aspx?space=1&num=1152

হ্যাপি কোডিং