Don’t Try to Count - Detailed Revision Notes

Problem: Codeforces 1881B | Rating: 800 | TLE CP-31 Sheet Problem #7

Problem Statement

Given a string `X` of length `n` and a string `S` of length `m` such that $n \times m \leq 25$, you can perform the following operation any number of times:

- Append the current `X` to itself (i.e., `X = X + X`).

After each operation, `X` changes to the new concatenated string. Determine the **minimum** number of operations required for `S` to appear as a **substring** of `X`. If it never appears, output `-1`.

Input Format:

- First line: `T` (number of test cases)
- For each test case:
  - Line: integers `n`, `m`
  - Line: string `X`
  - Line: string `S`

Output:

- For each test case, print the minimum operations or `-1` if impossible.

Key Intuition & Observations

1. **Upper Bound on Operations:** Since $n \times m \leq 25$, worst-case $n=1$ and $m=25$ (e.g., `X="a"`, `S="aaaa...a"` of length 25). Each operation doubles length of `X`:
   - After k operations, length = $n \times 2^k$.
   - To cover $m=25$, need smallest k such that $n \times 2^k \geq m$. For $n=1$, $2^k \geq 25 \rightarrow k=5$ (since $2^5=32$). Further operations only produce longer repeats of same pattern, so if not found by k=5, never found.
2. **Brute-Force Feasibility:** Maximum 6 versions of `X` (from 0 to 5 operations). Each check scans `X` for substring `S` in $O(|X| \times |S|)$ which is $O((n \times 2^k) \times m)$. At k=5, $n \times 2^5 \leq 32$, $m \leq 25 \rightarrow O(32 \times 25)=800$ operations per test. With $T \leq 10^4$, worst-case $\sim 8 \times 10^6$ operations fits within limits.

Solution Approach

1. **Precompute X variants:**
   - $X_0 = X$ (0 ops)
   - For i in [1..5]: $X_i = X_{i-1} + X_{i-1}$
2. **Check substring:**
   - For each i from 0 to 5:
     - If $|X_i| < m$, continue
     - Slide a window of size m over X_i and compare to S (or use built-in `find`).
     - If found, record i and break.
3. **Output:**
   - If found at $i \leq 5$, print i; otherwise print `-1`.

Time & Space Complexity

- **Time (per test):** $O(6 \times |X_5| \times m) = O((n \times 32) \times m) \leq O(32 \times 25) = 800 \rightarrow$ total $O(T \times 800)$.
- **Space:** $O(n \times 2^5)$ to store largest X_5, i.e. $O(32n)$.

Example Walkthroughs

Test Case	Operations	Answer
X=a, S=aaaaa (m=5)	X_0=a (len=1), X_1=aa (2), X_2=aaaa (4), X_3=aaaaaaaa (8): found at i=3	3
X=aba, S=abaaba	X_0=aba(3)<6, X_1=abaaba(6): found at i=1	1
X=xyz, S=xyzxyzxyz (m=9)	X_0=3<9, X_1=6<9, X_2=12: substrings include xyzxyzxyz at positions 0→8 → i=2	2
X=ab, S=c	None of X_0...X_5 contains 'c'	-1

Common Pitfalls

- **Infinite Loop Assumption:** Forgetting that doubling stops at 5 ops suffices by size constraints.
- **Off-by-One in Window:** Loop limit should be $i \leq |X_i| - m$ inclusive.
- **Ignoring $|X_i| < m$ check:** Skip checks when current X_i is shorter than S.

Competitive Programming Insights

- **Bounding Operations:** Use problem constraints ($n \times m \leq 25$) to cap brute force iterations.
- **Doubling Patterns:** Recognize exponential growth to meet length requirements quickly.
- **Window Sliding:** Efficient substring check via sliding window or built-in string search.

Final C++ Implementation Sketch

```
bool contains(const string &X, const string &S) {
    if (X.length() < S.length()) return false;
    return X.find(S) != string::npos;
}

int solve(string X, string S) {
    vector<string> V(6);
    V[0] = X;
    for (int i = 1; i <= 5; ++i)
        V[i] = V[i-1] + V[i-1];
    for (int i = 0; i <= 5; ++i) {
        if (contains(V[i], S)) return i;
    }
    return -1;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    int T; cin >> T;
    while (T--) {
        int n, m; cin >> n >> m;
        string X, S; cin >> X >> S;
        cout << solve(X, S) << '\n';
    }
    return 0;
}
```