My Short LaTeX Template Demo

Qilin Ye

January 13, 2024

In the following demo I will try to explain how to use my custom LaTeX template. I will assume that you know *basic* LaTeX syntax. This includes, for example, how to use math mode, or how to type subscripts, superscripts, sums, and so on. If you are not familiar with those, check out this extremely short guide on Overleaf.

Customized Theorem Environments

I have defined an extensive list of Theorem environments, including but not limited to Theorems/Corollaries, Lemmas, Definitions, Problems, and Proofs/Remarks/Solutions. Here are some examples.

Theorem 1.1

Bold *italicized* underlined `code` and `code again` WHATEVER THIS IS colorful a funny box $\mathcal{AND}$ $\mathbb{SO}$ $\mathfrak{DR}$.

Theorem

This is an unnumbered Theorem.

Theorem 1.2: Theorem with Name

This Theorem has a custom name.

Corollary 1.3

I have defined these environments so that Theorems, Corollaries, and Definitions share the same counter. If you would like to change this, look over the definitions in the preamble.

Problem 1

The Problem environment uses a separate counter for convenience. Usually you just want to start with "problem 1."

Now, for a few proofs:

Proof. This is a proof. □

A customized proof. This proof has a different name! □

Solution. Really, just another proof in disguise, except we don't want the QED symbol at the end.

Remark. Quite ugly color for a remark but I'm just not bothered to fix it.

Pseudocode and Algorithm Environments

Two examples:

Algorithm 1: Ford-Fulkerson Algorithm

Inputs: directed graph $G = (V, E)$ with edge capacities $c(e)$; source s, sink t

start with zero flow, i.e., $f(e) = 0$ for all edge
while residual graph $G(f)$ contains an $s - t$ path **do**
    let P be one such $s - t$ path
    `augment`(f, P), update flow
    `compute_residual_graph`$(G(f), f)$, update residual graph
Return: flow f

Function `compute_residual_graph`(*Graph* $G = (V, E)$, *flow* f)**:**
    start with empty edge set $G(f) = (V, \varnothing)$
    for *each edge* $e = (u, v) \in E$ **do**
        if $f(e) < c(e)$ **then** add (u, v) to $G(f)$ with capacity $c(e) - f(e)$
        if $f(e) > 0$ **then** add (v, u) [*reversed order*] to $G(f)$ with capacity $f(e)$
    Return: residual graph $G(f)$

Function `augment`(*flow* f, *path* P)**:**
    let ϵ be the smallest residual capacity along path P in $G(f)$
    for *each edge* $e = (u, v) \in P$ **do**
        if e *is a forward edge* **then**
            $f'(e) \leftarrow f(e) + \epsilon$
        else
            $f'(e) \leftarrow f(e) - \epsilon$ `// backward edge`
    Return: flow f'

```python
def foo(k, n):
    if k * n % 2 == 1: return 0

    # I have removed all comments so you don't know what the code does.
```

```
    # Even if you figured out, code has been deliberately altered in the wrong direction so don't attempt to copy!
    # It may take you more time to fix my code than just to come up with your own solution...
    max_mask = (1 << (k+1)) - 1
    dp = [[[0 for _ in range(1 << (k+2))] for _ in range(n+1)] for _ in range(k+1)] # padding
    dp[k][0][0] = 1
    for j in range(1, n+1):
        for mask in range(max_mask):
            dp[0][j][mask << 1] += dp[k][j-1][mask]
        for i in range(1,k+1):
            for mask in range(max_mask+1):
                NE, SW = mask & (1 << (i-1)), mask & (1 << i)
                if NE and SW: continue
                if NE and not SW: dp[i][j][mask ^ (1 << i)] += dp[i-1][j][mask]
            elif SW and not NE: dp[i][j][mask ^ (1 << (i-1))] += dp[i-1][j][mask]
                else:
                    dp[i][j][mask ^ (1 << (i-1))] += dp[i][j-1][mask]
                    dp[i][j][mask ^ (1 << i)] += dp[1][j-1][mask]
    return dp[k][n][0]
```

Some Advice on Typesetting Large Equations

There's not much customization going on here, but here are some useful tips to greatly increase readability when you are dealing with big equations:

- **Use colors to highlight important parts.** (Don't overuse.)

  By equicontinuity there exists $\delta > 0$ such that

  $$|g_n(x) - g_n(y)| < \epsilon/3 \qquad \text{for all } n \in \mathbb{N} \text{ if } d(x,y) < \delta. \tag{1}$$

  ... by Cauchy condition

  $$|g_n(x_i) - g_m(x_i)| < \epsilon/3 \qquad \text{for all } m, n \geqslant N \text{ and } x_i \in \{x_1, \cdots, x_k\}. \tag{2}$$

  Finally, using (1) twice and (2) once, for $m, n \geqslant N$ and $x \in X$ we obtain

  $$|f_m(x) - f_n(x)| \leqslant |f_m(x) - f_m(x_i)| + |f_m(x_i) - f_n(x_i)| + |f_n(x_i) - f_n(x)| < \frac{\epsilon}{3} + \frac{\epsilon}{3} + \frac{\epsilon}{3} = \epsilon.$$

- **Use underbraces and overbraces when there are many terms.**

  $$I[f] := \frac{1}{2} \int_0^1 \left(\underbrace{(f'(x))^2}_{\substack{\text{energy of} \\ \text{the system}}} + \underbrace{V(f(x))}_{\text{potential}} \right) \mathrm{d}x$$

- **When using multi-line `align` (or similar environments), add per-line comments for additional clarity.** *(The next example also contained hyperlinks. I have created some dummy labels here to make the hyperlink boxes*

appear.)

$$f(x) = \sum_{k=0}^{n-1} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k + \begin{cases} \dfrac{f^{(n)}(\xi)}{n!}(x-x_0)^n & \text{Lagrange form, Theorem 10.4} \\ o\big((x-x_0)^{n-1}\big) & \text{Peano form, Theorem 10.7} \\ \displaystyle\int_{x_0}^{x} f^{(n)}(t)\frac{(x-t)^{n-1}}{(n-1)!}\,\mathrm{d}t & \text{integral form, Theorem 12.3.} \end{cases}$$

$$\begin{aligned} |S_x(t) - S_y(t)| &= \left| \left(x_0 + \int_0^t f(x(s))\,\mathrm{d}s\right) - \left(x_0 + \int_0^t f(y(s))\,\mathrm{d}s\right) \right| \\ &= \left| \int_0^t [f(x(s)) - f(y(s))]\,\mathrm{d}s \right| && \text{(by Lemma 11.4.1)} \\ &\leqslant \int_0^t \big|f(x(s)) - f(y(s))\big|\,\mathrm{d}s && \text{(by Theorem 11.5)} \\ &\leqslant \int_0^t L|x(s) - y(s)|\,\mathrm{d}s && \text{(by Lipschitz assumption)} \\ &\leqslant \int_0^t L\|x-y\|_{\sup}\,\mathrm{d}s = \|x-y\|_{\sup} \cdot Lt \leqslant \|x-y\|_{\sup} \cdot LT. && \text{(by Lemma 11.4.2 \& .6)} \end{aligned}$$

Some Very Helpful Links

(1) ChatGPT. I learned LaTeX before its advent. I envy you all. (It still fails at drawing Tikz diagrams, though.)

(2) Overleaf LaTeX guide. Covers almost all basic stuff.

(3) TeX StackExchange. Do I even need to explain?

(4) Mathpix snipping tool. Extremely powerful OCR, especially when you get lazy and don't want to type a chunk of equations from a textbook, for example.

(5) Mathcha online editor. Quite useful place to draw diagrams and export to Tikz, for those who do not want to directly write Tikz code in LaTeX.