Twilight: Adaptive Attention Sparsity with Hierarchical Top-p Pruning

Chaofan Lin, **Jiaming Tang, Shuo Yang, Hanshuo Wang, Tian Tang, Boyu Tian, Ion Stoica, Song Han, Mingyu Gao**

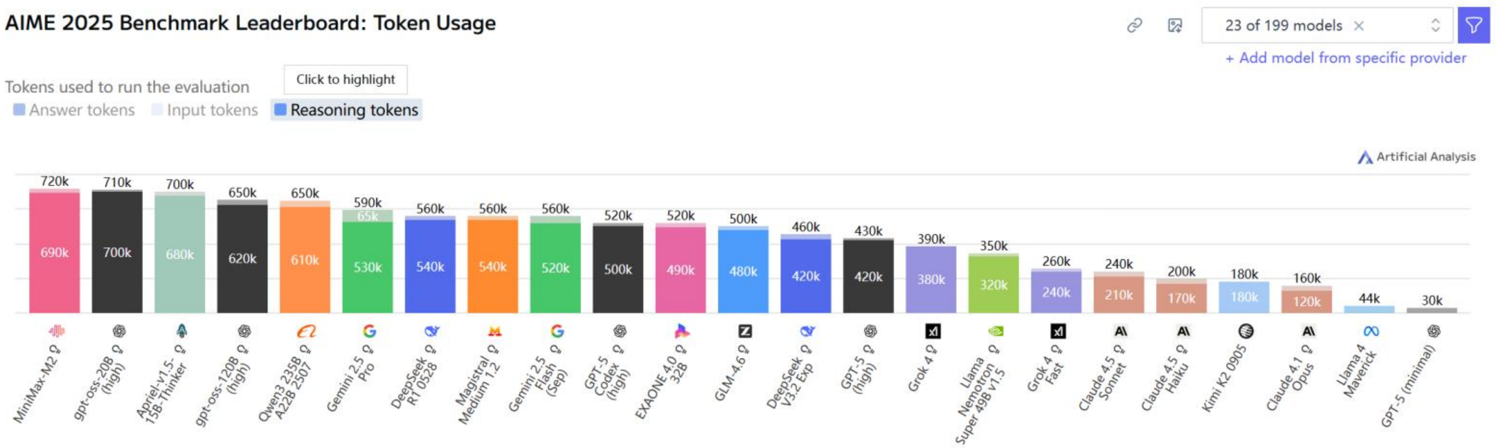
Tsinghua University
Massachusetts Institute of Technology
University of California, Berkeley

https://github.com/tsinghua-ideal/Twilight

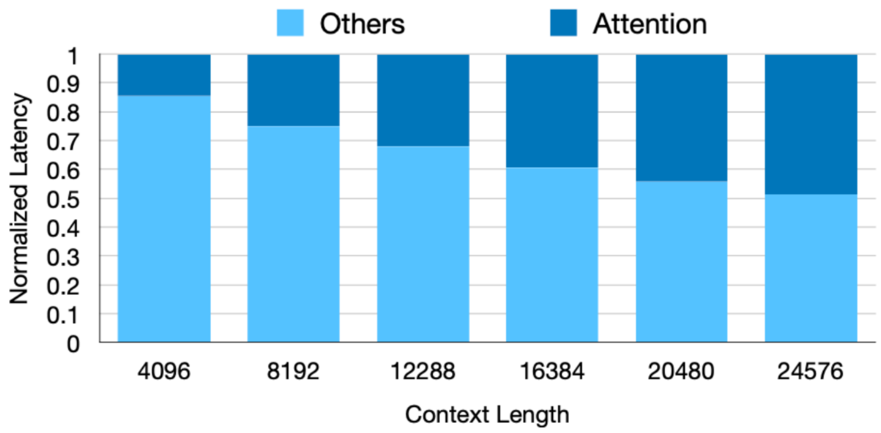
Background

Long-context LLMs are powerful but computationally expensive

- Trend: Long Context windows are becoming the new standard for state-of-the-art LLMs, especially for reasoning models.
- For long context LLM inference, attention dominates the latency.



Reasoning tasks like AIME cost nearly 1M tokens

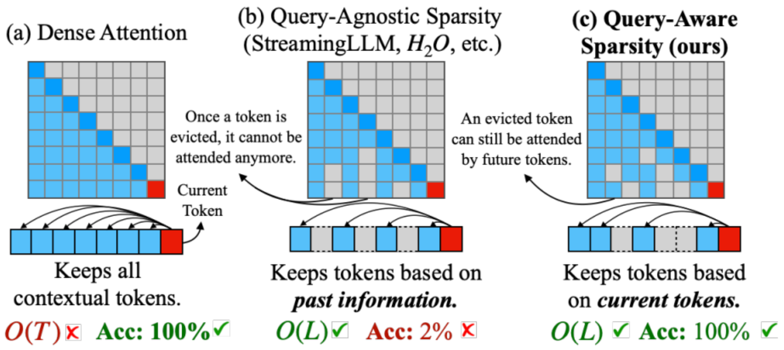


Attention becomes the bottleneck as the sequence length increases

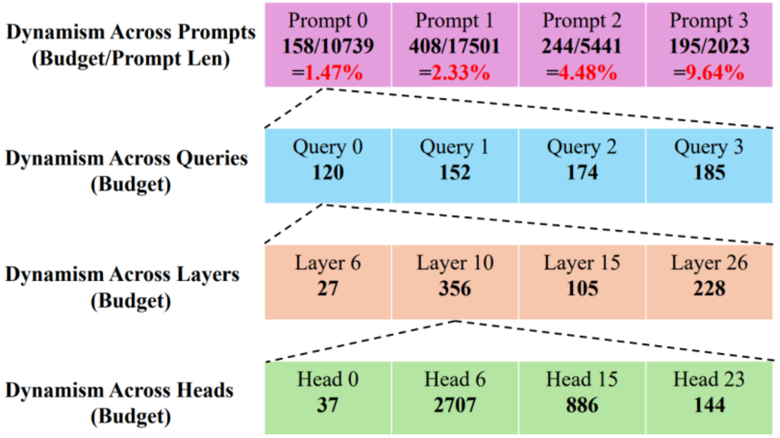
Background

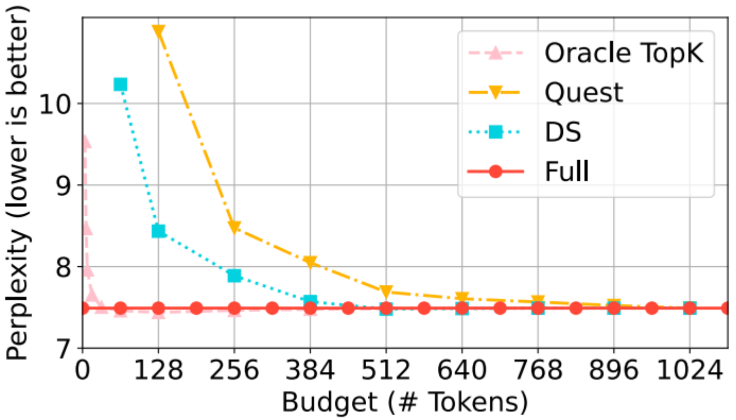
Top-k Sparse Attention to reduce KV cache loading

- Since attention is memory-bound, previous works propose sparse attention, which first estimates attention scores then selectively loads only important tokens.
- However, the main challenge of top-k sparse attention is to find a universally applicable budget to all scenarios.



Previous work (Quest)





The best budget choices vary dynamically across different levels.

Quest: Query-Aware Sparsity for Efficient Long-Context LLM Inference. Tang et al.

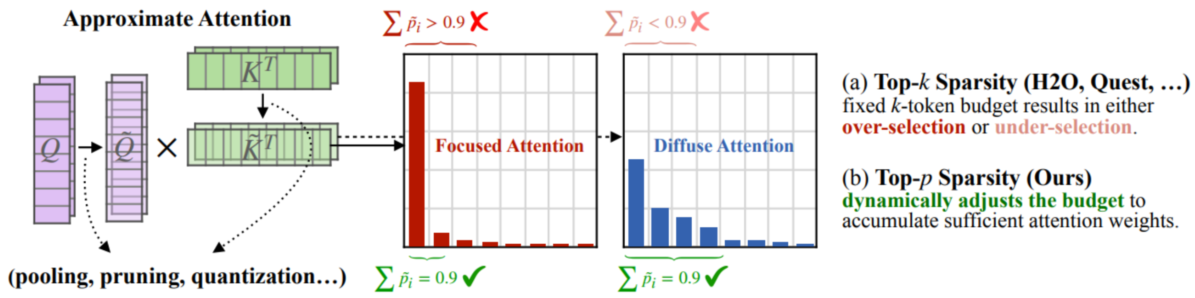
Bringing Top-p Sampling to Sparse Attention

Top-p Sparse Attention is inherently budget-adaptive

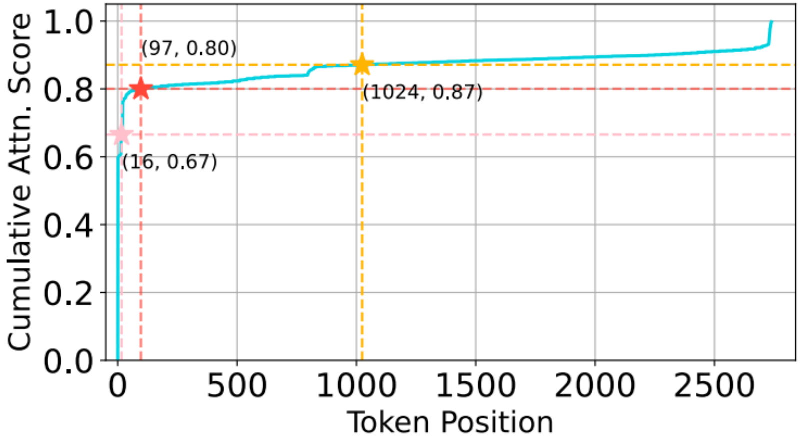
- We argue that the core reason for budget dynamism is the dynamic nature of the attention weight distributions at runtime, thus propose Top-p Sparse Attention.

Definition 3.3 (Oracle Top-p Sparse Attention). Given the threshold p,

$$\mathcal{I} = \arg\min_{\mathcal{I}} |\mathcal{I}| \quad \text{s.t.} \sum_{i\in\mathcal{I}} \mathbf{W}[i] \geq p$$



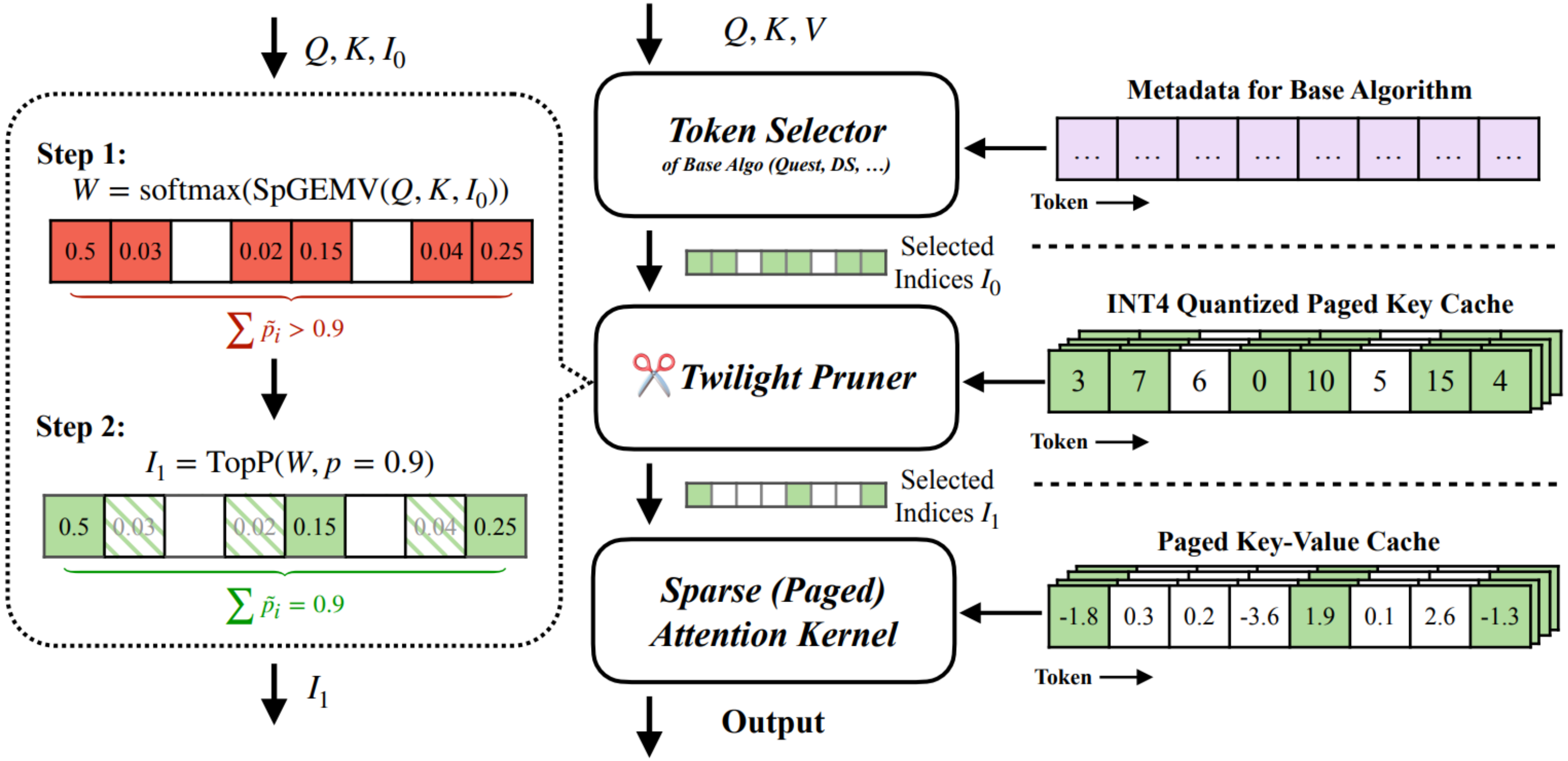
Diverse distributions observed in attention weights of different attention heads.



Cumulative attention scores of different budget selections in one example attention head.

Twilight

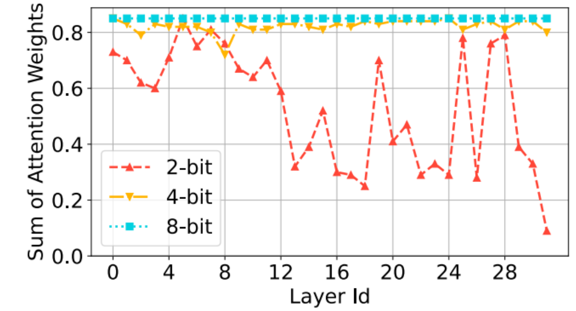
- **Key Design:** Hierarchical Select-then-Prune architecture as a unified optimizer for all existing top-k based sparse attention methods (denoted as *BaseAlgo*).
- First *BaseAlgo* uses a conservative, relatively large budget. Then **Twilight** further prunes them using efficient top-p Pruner.

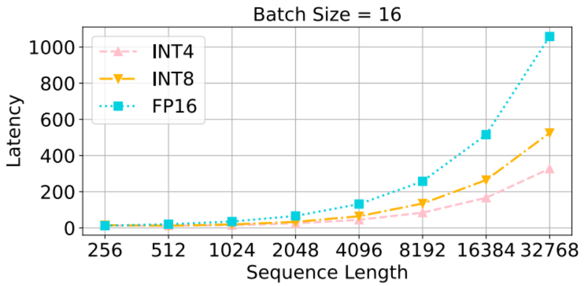


Twilight

To Achieve the Efficient Pruner

- Efficient SpGEMV with 4-bit Quantization of Key Cache to estimate token importance: we find that 4-bit strikes a balance between accuracy and efficiency.
- Efficient Sorting-free Top-p via binary search modified from FlashInfer.
- Load Balancing with Awareness of Head Dynamism with GQA adaption.

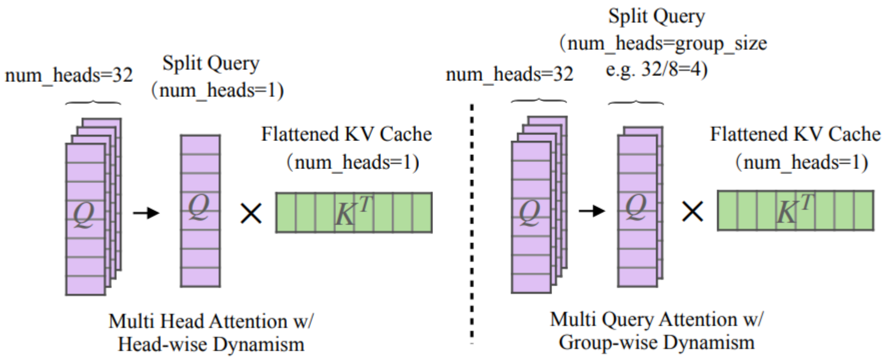




Algorithm 1 Top-p via Binary Search.

Input: normalized attention weights $W \in \mathbb{R}^{BS \times H \times N}$, top-$p$ threshold p, hyper-parameter ϵ.
Output: indices $\mathcal{I}$, mask $\mathcal{M} \in \{0,1\}^{BS \times H \times N}$.

$l = 0, r = \texttt{max}(W), m = (l + r)/2;$
repeat
  $W_0 = \texttt{where}(W < m,\ 0.0,\ W);$
  $W_1 = \texttt{where}(W \leq l,\ \text{INF},\ W);$
  $W_2 = \texttt{where}(W > r,\ -\text{INF},\ W);$
  if $\texttt{sum}(W_0) \geq p$ **then**
    $l = m;$
  else
    $r = m;$
  end if
until $\texttt{max}(W_2) - \texttt{min}(W_1) \geq \epsilon$
Select indices $\mathcal{I}$ and set mask $\mathcal{M}$ where $W \geq l$;
return $\mathcal{I}, \mathcal{M}$;



FlashInfer: Efficient and Customizable Attention Engine for LLM Inference Serving. Ye et al.

Accuracy Evaluation

- Twilight achieves nearly **no accuracy loss** on three medium-context benchmarks and two long-context benchmarks (LongBench, RULER).

Table 2: Average scores on 12 different tasks from Longbench. We report relative error changes (improvement or degradation) when integrating Twilight with each base algorithm. Detailed results are in Table 5 in Appendix C.

	Budget	Longchat-7B -v1.5-32k	LLaMA-3.1-8B -Instruct
Full	32k	36.78	52.01
	Twilight	38.52 (+4.7%)	51.64 (-0.7%)
MagicPIG	K=8, L=75	-	51.70
	K=10, L=150	-	51.32
Quest	256	31.26	38.20
	1024	36.85	47.79
	4096	37.33	50.79
	8192	37.10	51.44
	Twilight	38.04 (+2.5%)	51.57 (+0.3%)
DS	256	35.32	45.74
	1024	35.96	49.43
	4096	36.31	50.98
	8192	36.62	51.14
	Twilight	**38.71** (+5.7%)	**51.73** (+1.2%)

Table 3: Average scores on RULER.

	Budget	16k	32k	64k	96k	Avg.
Full	100%	92.88	89.42	85.17	85.23	88.18
	Twilight	93.13	89.10	84.64	83.10	87.49
MagicPIG	K=8, L=75	92.22	**89.37**	84.07	82.58	87.06
	K=10, L=150	91.38	88.20	83.34	82.02	86.23
Quest	4%	79.35	79.8	78.64	73.22	77.75
	8%	87.31	83.06	80.82	75.28	81.62
	Twilight	91.53	87.97	84.12	**82.96**	86.65
DS	4%	92.04	88.11	84.43	82.56	86.79
	8%	92.89	88.70	84.39	82.72	87.18
	Twilight	**93.54**	89.24	**85.91**	82.81	**87.88**

Table 4: Results on 3 medium-context benchmarks.

	GSM8K(flexible/strict)↑	COQA(em/f1)↑	PG-19 Perplexity↓
LLaMA-2-7B-Chat			
Full	0.2290/0.2282	0.5935/0.7511	7.503
Quest	0.0523/0.0508	0.5710/0.7425	14.15
DS	0.2191/0.2190	0.5855/0.7401	7.622
Twilight	**0.2153/0.2115**	**0.6088/0.7642**	**7.600**
(Twilight Avg. Budget)	90.82	91.86	102.58
LLaMA-3.1-8B-Instruct			
Full	0.7726/0.7475	0.6363/0.7882	7.490
Quest	0.3639/0.3533	0.6007/0.7554	19.00
DS	0.6194/0.6027	**0.6455/0.7964**	7.967
Twilight	**0.7771/0.7604**	0.6325/0.7869	**7.529**
(Twilight Avg. Budget)	112.40	86.85	110.98

Efficiency Evaluation

- Twilight accelerates self-attention operator by 2.4× (FlashInfer) and 1.4× (Quest) at batch size=64
- And for E2E per-token latency, Quest-Twi is 3.9× compared to FlashInfer and 1.35× to Quest at batch size=256.

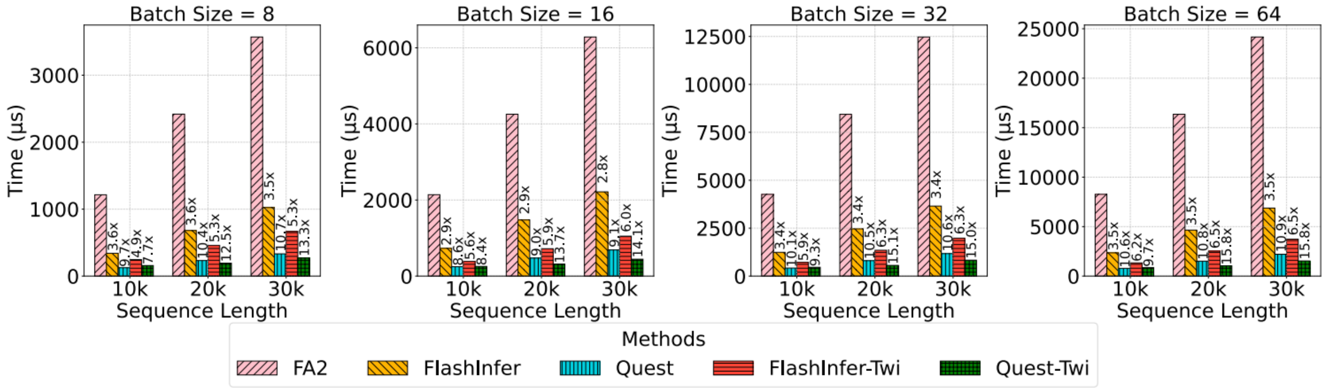


Figure 7: Latencies and speedups of self-attention at different sequence lengths and batch sizes.

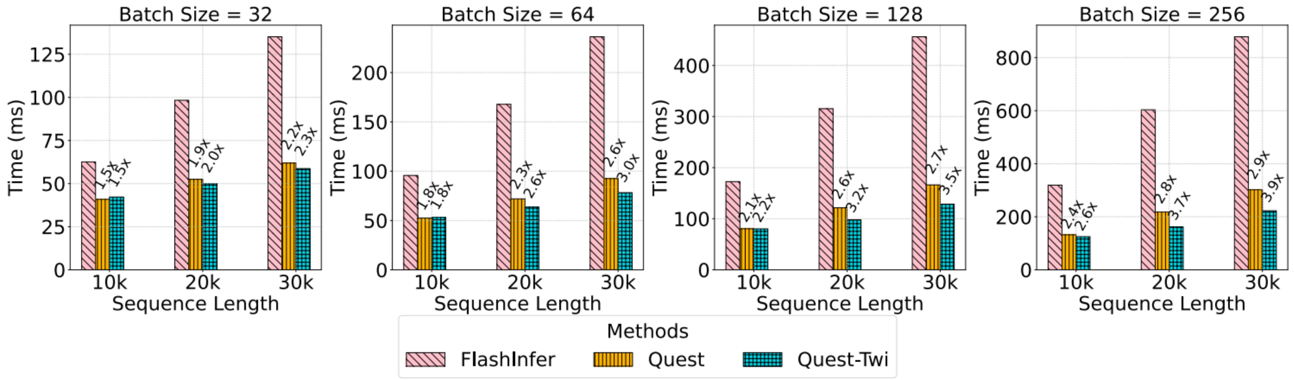


Figure 8: Time-Per-Output-Token (TPOT) improvements in end-to-end serving scenarios.

- We propose Twilight, a composable optimizer to accelerate any existing top-k sparse decoding methods through **hierarchical top-p pruning**, making them **efficient and budget-adaptive**.
- Paper: https://arxiv.org/abs/2502.02770
- Code: https://github.com/tsinghua-ideal/Twilight

Thanks for Listening

Chaofan Lin
lcf24@mails.tsinghua.edu.cn

清華大學
Tsinghua University