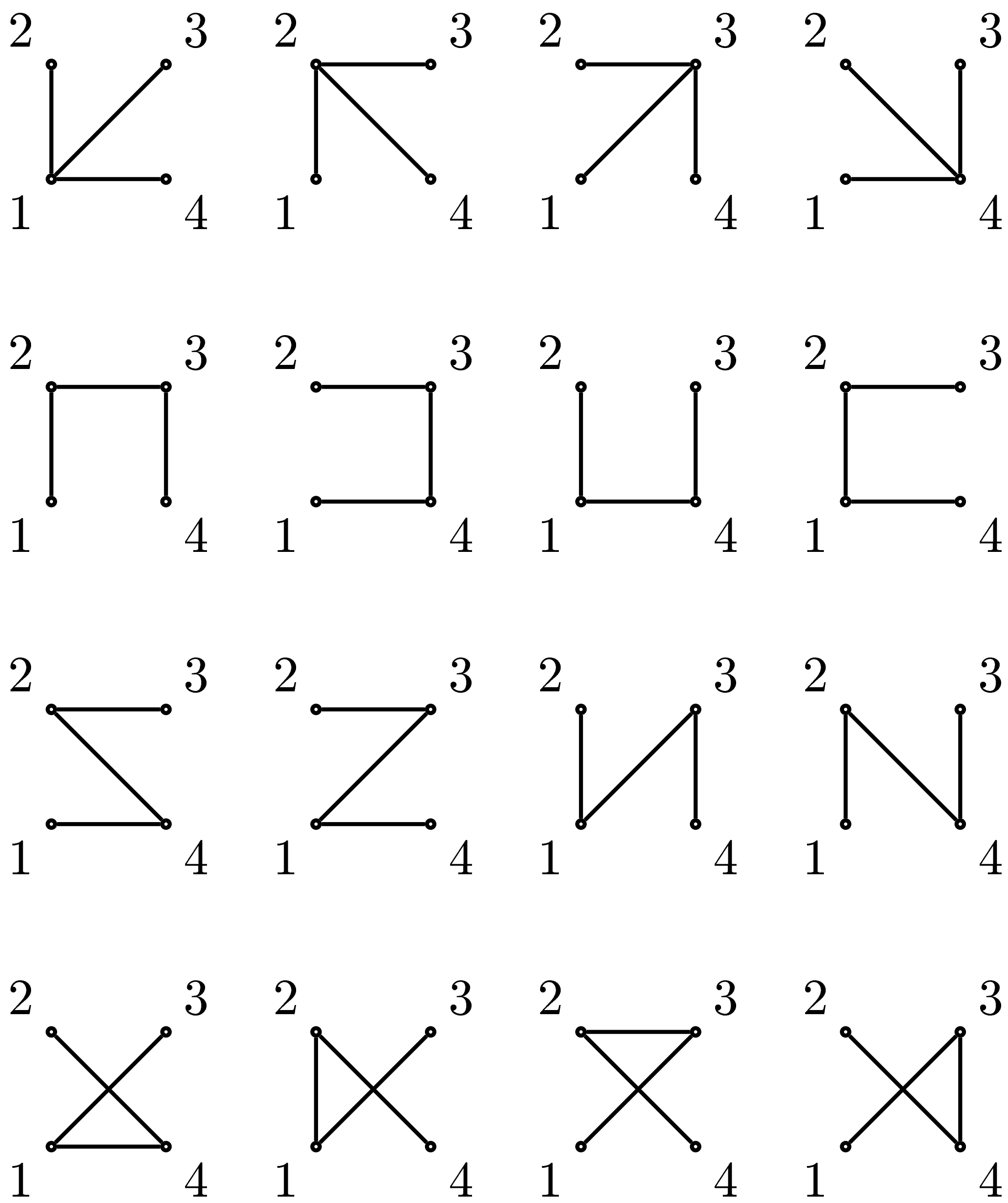
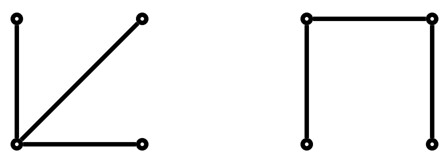


Alle (markierten) Bume mit vier Knoten.



Alle nicht-isomorphen Bume auf 4 Knoten

1. Schritt	$i = 1, S = \emptyset,\ t_1 \dots t_{n-2} = 3873773$ $\Longrightarrow \quad s_1 = 1, e_1 = \{1, 3\}$
2. Schritt	$i = 2, S = \{1\},\ t_2 \dots t_{n-2} = 873773$ $\Longrightarrow \quad s_2 = 2, e_2 = \{2, 8\}$
3. Schritt	$i = 3, S = \{1, 2\},\ t_3 \dots t_{n-2} = 73773$ $\Longrightarrow \quad s_3 = 4, e_3 = \{4, 7\}$
4. Schritt	$i = 4, S = \{1, 2, 4\},\ t_4 \dots t_{n-2} = 3773$ $\Longrightarrow \quad s_4 = 5, e_4 = \{5, 3\}$
5. Schritt	$i = 5, S = \{1, 2, 4, 5\},\ t_5 \dots t_{n-2} = 773$ $\Longrightarrow \quad s_5 = 6, e_5 = \{6, 7\}$
6. Schritt	$i = 6, S = \{1, 2, 4, 5, 6\},\ t_6 \dots t_{n-2} = 73$ $\Longrightarrow \quad s_6 = 8, e_6 = \{8, 7\}$
7. Schritt	$i = 7, S = \{1, 2, 4, 5, 6, 8\},\ t_7 \dots t_{n-2} = 3$ $\Longrightarrow \quad s_7 = 7, e_7 = \{7, 3\}$
8. Schritt	$S = \{1, 2, 4, 5, 6, 8\}$ $e_8 = \{9, 3\}$

BAUM→PRFERCODE

Eingabe: Baum $T = ([n], E)$ mit Knotenmenge $[n]$

Ausgabe: Wort $t_1 \dots t_{n-2}$ ber dem Alphabet $[n]$

$i \leftarrow 1$

while Anzahl Knoten $|V| > 2$ **do begin**

bestimme kleinstes Blatt v im Baum T

$t_i \leftarrow$ Nachbar von v im Baum T

entferne v aus T

$i \leftarrow i + 1$

end

Prfercode→Baum

Eingabe: Wort $t_1 \ldots t_{n-2}$ ber dem Alphabet $[n]$

Ausgabe: Baum $T = ([n], E)$

$S \leftarrow \emptyset$

for $i \leftarrow 1$ **to** $n - 2$ **do begin**

whle kleinsten Knoten $s_i \in [n] \setminus S$, der nicht in $t_i \ldots t_{n-2}$ vorkommt

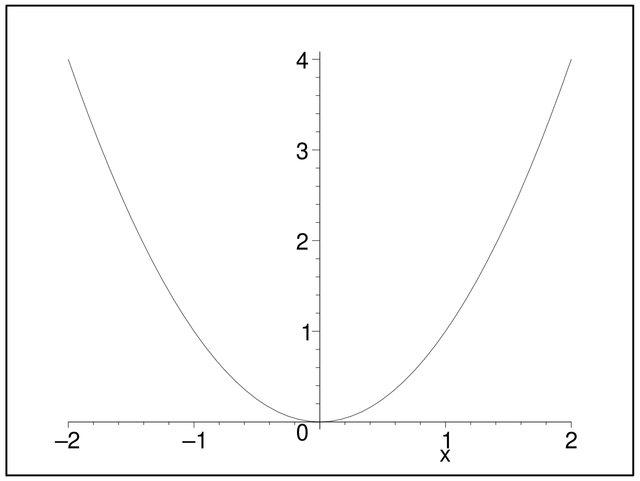
fge die neue Kante $e_i := \{s_i, t_i\}$ in den Graphen ein

$S \leftarrow S \cup \{s_i\}$

end

fge die Kante $e_{n-1} := [n] \setminus S$ in den Graphen ein

```
> plot( x^2, x=-2..2);
```




```
> int(x^2,x);
```

$$\frac{1}{3}x^3$$

```
> int(x^2,x=-2..2);
```

$$\frac{16}{3}$$

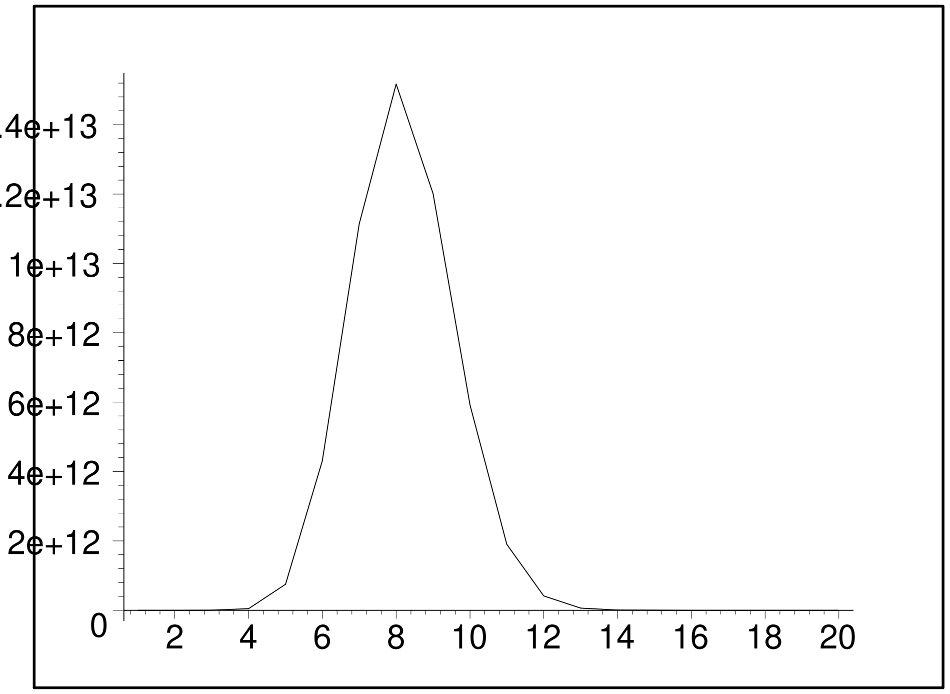
```
> with(plots, listplot);
```

$$[listplot]$$

```
> with(combinat, stirling2);
```

$$[stirling2]$$

```
>
listplot([seq(stirling2(20,k),k=1..20)]);
```



```
> with(networks);
```

[*acycpoly*, *addedge*, *addvertex*, *adjacency*, *allpairs*, *ancestor*, *arrivals*, *bicomponents*, *charpoly*, *chrompoly*, *complement*, *complete*, *components*, *connect*, *connectivity*, *contract*, *countcuts*, *counttrees*, *cube*, *cycle*, *cyclebase*, *daughter*, *degreeseq*, *delete*, *departures*, *diameter*, *dinic*, *djspantree*, *dodecahedron*, *draw*, *duplicate*, *edges*, *ends*, *eweight*, *flow*, *flowpoly*, *fundcyc*, *getlabel*, *girth*, *graph*, *graphical*, *gsimp*, *gunion*, *head*, *icosahedron*, *incidence*, *incident*, *indegree*, *induce*, *isplanar*, *maxdegree*, *mincut*, *mindegree*, *neighbors*, *new*, *octahedron*, *outdegree*, *path*, *petersen*, *random*, *rank*, *rankpoly*, *shortpathtree*, *show*, *shrink*, *span*, *spanpoly*, *spantree*, *tail*, *tetrahedron*, *tuttepoly*, *vdegree*, *vertices*, *void*, *vweight*]

```
> G :=
graph({1,2,3,4,5},{{1,2},{1,3},{4,5}}):edges(G);
```

$$\{e3,\ e2,\ e1\}$$

```
> connectivity(G);
```

$$0$$

```
> draw(G);
```